ИНФОРМАТИКА, ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И УПРАВЛЕНИЕ

УДК 519.2:004.421.5:004.7

В. Н. ЗАДОРОЖНЫЙ

Омский государственный технический университет

ПОВЫШЕНИЕ ТОЧНОСТИ GPSS-МОДЕЛЕЙ ПУТЕМ ПРИМЕНЕНИЯ ГЕНЕРАТОРА СЛУЧАЙНЫХ ЧИСЕЛ «ВИХРЬ МЕРСЕННА»

Рассматриваются подходы к применению в GPSS World внешнего генератора случайных чисел. Приводятся примеры применения генератора, основанного на алгоритме «Вихрь Мерсенна», для моделирования систем с очередями. Демонстрируются появляющиеся при этом возможности существенного повышения точности результатов моделирования.

Ключевые слова: имитационное моделирование, генератор случайных чисел, системы массового обслуживания, распределения с тяжелыми хвостами.

1. Введение. Несмотря на широкое распространение множества разнообразных систем имитационного моделирования, система GPSS World [1] до сих пор является одной из самых популярных. Для многих задач моделирования язык GPSS является идеальным средством описания моделей и выполнения имитационных экспериментов. Привлекают также достаточно высокие скорость и точность выполнения большинства вычислительных процессов, реализуемых на GPSS. Вместе с тем естественная прежде упрощенная (по современным меркам) реализация генераторов случайных чисел (ГСЧ), встроенных в GPSS World, становится в настоящее время препятствием для повышения точности результатов моделирования за счет увеличения объемов выборок до больших размеров, вполне достижимых на современных персональных компьютерах (ПК) средней мощности.

Как показывают эксперименты (которые нетрудно воспроизвести и проверить), ГСЧ класса

```
GENERATE  1
SAVEVALUE XN1,(MtRand())
TERMINATE 1

PROCEDURE MtRand() BEGIN
TEMPORARY Z;
Z = Call("mtrand.dll", "?nextNum@@YAHXZ");
Z = Z/2147483647;
RETURN (Z);
END;
```

Рис. 1. GPSS-программа с вызываемой из нее процедурой MtRand()

```
PROCEDURE MtInit(Arg) BEGIN
Call_Integer("mtrand.dll", "?init@@YAHH@Z", Arg);
END;
```

Рис. 2. Процедура инициализации генерируемой псевдослучайной последовательности

Uniform(j,0,1), встроенные в GPSS World [1], выдают последовательности, которые начинают самоповторяться, начиная с 2 048 000 000-го числа. При разных j эти ГСЧ выдают одну и ту же (сдвинутую на разные фазы) периодическую последовательность с длиной периода 2 047 999 999. Это не позволяет в экспериментах с большой длиной выборки использовать разные ГСЧ как независимые. На практике, как показывают эксперименты, результаты моделирования (например, оценка среднего времени ожидания заявок в очереди) могут при совместном использовании двух таких ГСЧ существенно смещаться уже при 1 млн обращений к ним. При объемах выборок порядка 1 млн разные ГСЧ начинают «интерферировать», и результаты моделирования существенно искажаются [2]. Поэтому в серьезных имитационных экспериментах на GPSS World (с длинными прогонами модели) использовать более одного ГСЧ в одной модели не приходится, ведь прогон модели с 1 млн обращений к ГСЧ занимает на ПК не более нескольких секунд.

Генераторы класса RNj в GPSS World более разнообразны. Длина периода генераторов RN1 и RN2 составляет 3 145 728 304. Но эти генераторы при совместном использовании тоже могут «интерферировать». Так, начиная с 811 795 903-го числа последовательность RN1 совпадает с началом последовательности RN2. У других ГСЧ класса RNj длина периода может быть другой, но автор не нашел таких ГСЧ, у которых она больше, чем у ГСЧ RN1.

В целом все найденные длины периодов недостаточны для реализации достаточно больших объемов выборок. В статье предлагается способ корректного повышения объемов выборок, доступных в GPSS World, основанный на подключении внешнего датчика. Показано, что это позволяет при моделировании систем массового обслуживания (СМО), описываемых распределениями с тяжелыми хвостами (РТХ), значительно расширить возможности решения возникающих здесь проблем [3].

2. ГСЧ MtRand. В качестве внешнего генератора псевдослучайных чисел в моделях на GPSS World в настоящее время наиболее целесообразно использовать, по всей видимости, так называемый МТ-генератор, основанный на алгоритме «Вихрь Мерсенна» [4]. В Интернете без особого труда можно найти доступный код этого генератора на языке С++. Из этого кода, для того чтобы генератор можно было использовать в GPSS World, нужно скомпилировать функцию (файл) *.dll, поддерживающую протокол CDECL, и поместить ее в папку GPSS World или в папку с выполняемыми моделями *.gps. Такую функцию можно вызывать из PLUS-процедур [1], используя соответствующий формат команды вызова с указанием имени исполняемого файла и имени функции.

В разработках и исследованиях, описанных ниже, использовался код С++ наиболее распространенной 32-битной версии МТ-генератора, обеспечивающей длину периода $2^{19\,937} - 1 \approx 10^{6000}$. Полученная его компиляцией dll-функция, помещенная затем в папку GPSS World, названа mtrand.dll. При вызове этой функции она возвращает целое псевдослучайное число, равномерно распределенное в интервале от 0 до $(2^{31} - 1)$. На рис. 1 приведен пример программы с вызовом PLUS-процедуры MtRand(), вызывающей, в свою очередь, функцию mtrand.dll для генерации стандартных случайных чисел.

PLUS-процедура MtRand() реализует здесь стандартную псевдослучайную величину (базовую случайную величину — БСВ), равномерно распределенную в интервале от 0 до 1 с шагом $\varepsilon = 1/(2^{31} - 1)$. Такой шаг ε с запасом обеспечивает девять точных десятичных цифр БСВ после запятой. Это заметно повышает точность реализации РТХ (критичную к точности представления БСВ, [5]), так как обычно в GPSS реализуется БСВ только с шестью цифрами после запятой.

В целом программа, представленная на рис. 1, при запуске ее (после трансляции) командой START $\langle n \rangle$ генерирует n значений БСВ и записывает их в первые n ячеек SAVEVALUE.

Скорость генерации псевдослучайных чисел процедурой MtRand() приблизительно в 2,5 раза меньше, чем скорость стандартной для GPSS World процедуры Uniform. Однако на фоне всех затрат времени на имитационное моделирование это, как правило, приводит к относительно небольшому замедлению. Повысить скорость процедуры MtRand() можно путем применения более быстрых программ, реализующих МТ-генератор. Сообщается, что среди них существуют такие, которые в 2–3 раза превосходят по скорости конгруэнтные генераторы [4], в том числе, следовательно, и генератор Uniform.

Использование MtRand() в каскадных методах реализации РТХ [3, 6–8] также ведет к повышению точности получаемых результатов. Кроме того, сверхбольшой период МТ-генератора делает возможным корректное, практически неограниченное увеличение объема получаемых выборок, что весьма актуально при моделировании СМО с РТХ.

3. Инициализация ГСЧ MtRand. Если нужно обеспечить повторную воспроизводимость генерируемой последовательности БСВ, то функцию mtrand.dll можно инициализировать «семенем» (целым положительным числом) с помощью PLUS-процедуры MtInit(s), представленной на рис. 2.

Процедура Mtinit(s) обращается к программе mtrand.dll и передает ей посредством аргумента Arg целое положительное число s для инициализации генератора. Если в разных экспериментах инициализировать генератор одним и тем же числом, то он выдает в них одну и ту же последовательность чисел.

Например, если функция mtrand.dll инициализируется семенем 1, то 10 чисел, формируемых программой (рис. 1) при ее запуске командой START

```
PROCEDURE MtExponential(Arg1,Arg2) BEGIN
TEMPORARYZ, XI;
STEP1: Z = Call("mtrand.dll", "?nextNum@@YAHXZ");
IF (Z = 0)THEN GOTO STEP1;
Z = Z/2147483647;
XI = -Arg2 # log(Z) + Arg1;
RETURN (XI);
END;
```

Рис. 3. Процедура реализации экспоненциальной случайной величины

```
PROCEDURE MtPareto(Arg1,Arg2) BEGIN
TEMPORARY Z, XI;
STEP1: Z = Call("mtrand.dll", "?nextNum@@YAHXZ");
IF (Z = 0)THEN GOTO STEP1;
Z = Z/2147483647;
XI = Arg1#Z^(-1/Arg2);
RETURN (XI);
END;
```

Рис. 4. Процедура реализации с.в. с распределением Парето

```
GENERATE (MtExponential(0,1))
QUEUE    1
SEIZE    1
DEPART   1
ADVANCE  (MtExponential(0,0.9))
RELEASE  1
TERMINATE

GENERATE 1000000
TERMINATE  1

PROCEDURE MtExponential(Arg1,Arg2) BEGIN
TEMPORARY Z, XI;
STEP1: Z = Call("mtrand.dll", "?nextNum@@YAHXZ");
IF (Z = 0)THEN GOTO STEP1;
Z = Z/2147483647;
XI = -Arg2 # log(Z) + Arg1;
RETURN (XI);
END;
```

Рис. 5. Модель M/M/1-системы, использующая MT-генератор

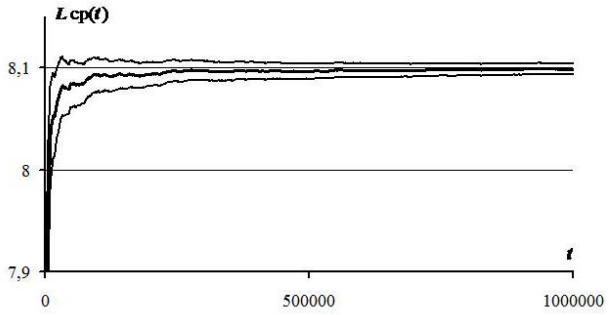

Рис. 6. ПП $L_{cp}(t)$ (средняя кривая). Верхняя и нижняя кривые ограничивают доверительную полосу, построенную по правилу трех сигм

10, таковы: 0.834044, 0.994370, 0.440649, 0.865115, 0.000229, 0.256249, 0.604665, 0.998081, 0.293512, 0.472178 (здесь числа округлены до 6 цифр после точки).

Если вызвать GPSS World N раз подряд, то откроется N окон, и процессы моделирования, запускаемые в этих окнах, будут выполняться на многоядерном компьютере параллельно, по одному на каждом ядре, практически без замедления (при числе ядер, не меньшем N). При использовании процедуры MtRand() псевдослучайные последовательности в каждом из окон генерируются своей уникальной копией функции mtrand.dll, инициализируемой (или не инициализируемой) независимо от остальных копий.

Для того чтобы использовать MT-генератор в GPSS World было более удобно, целесообразно написать библиотеку соответствующих PLUS-процедур, реализующих на основе MT-генератора разнообразные типовые вероятностные распределения.

4. Примеры реализации случайных величин с типовыми распределениями. Возможными значениями на выходе процедуры MtRand() являются как 0, так и 1. Чтобы исключить значение 1, можно в MtRand() величину Z нормировать делением на 231, а не на $(2^{31}-1)$, как на рис. 1. Для исключения нуля требуется дополнительная проверка, как это делается в представленной на рис. 3 процедуре MtExponential(s,m), реализующей экспоненциальную случайную величину (с.в.) с математическим ожиданием (м.о.) m и смещением s.

Процедура MtPareto(K,α) реализует с.в. с распределением Парето, где $K \geq 0$ — наименьшее значение (масштабный параметр), $\alpha > 0$ — параметр формы. Процедура имеет следующий вид (рис. 4).

В этой процедуре масштабный параметр K задается аргументом Arg1, параметр формы α — аргументом Arg2.

Оба приведенных примера генерации с.в. (рис. 3 и рис. 4) используют широко распространенный метод обратного преобразования БСВ (т.е. метод обращения функции распределения [9]).

5. Пример реализации и обработки выборки большого объема. Продемонстрировать новые возможности моделирования на GPSS World, приобретаемые благодаря использованию MT-генератора «Вихрь Мерсенна», можно на простом примере моделирования системы M/M/1 (простейшей СМО). Модель СМО, использующая MT-генератор посредством обращения к процедуре MtExponential(s,m) (рис. 3), представлена на рис. 5 (процедура транслируется вместе с использующей ее программой).

На рис. 6 показан график переходного процесса (ПП) $L_{cp}(t)$ — оценки для средней длины очереди в этой СМО. Значения ПП $L_{cp}(t)$ получены усреднением 10 000 независимых реализаций процесса изменения в модельном времени t системного числового атрибута QA1. Моделирование проводилось на интервале от 0 до 1 млн единиц времени. Точки отсчета (значения t, в которых фиксировались значения QA1) следовали с шагом 1000 единиц времени. Так как в этой простейшей СМО интенсивность поступления заявок λ равна 1, и среднее время их обслуживания b равно 0,9 (рис. 5), точное значение стационарной средней длины очереди L составляет $(\lambda b)^2/(1-\lambda b) = 8{,}1$ [10]. Весь эксперимент на ПК занял около 1,5 часа времени.

Как видно из рис. 6, ПП $L_{cp}(t)$ сходится к точному стационарному значению $L = 8{,}1$.

Значение $L_{cp}(t)$, достигнутое на момент 1 млн единиц модельного времени, составило 8,0987, близкое к 8,1. Допустимое правилом трех сигм отклонение составило для $L_{cp}(t)$ на этот момент времени величину $3\times0{,}00174$. В пересчете на одну реализацию процесса QA1(t) это составляет $3\times0{,}174 \approx 0{,}52$. MT-генератор инициализирован в этом эксперименте семенем 1. Общее число обращений к MT-генератору составило приблизительно $2\times10^6\times10^4 = 20$ млрд, что практически в 10 раз

```
        GENERATE (MtPareto(1,1.25))
VXOD    TEST L   Q1,25,OTKAZ
        QUEUE    1
        SEIZE    1
        DEPART   1
        ADVANCE  (MtPareto(1,1.5))
        RELEASE  1
        TERMINATE
OTKAZ   TERMINATE
        GENERATE 1000000
        TERMINATE              1

        PROCEDURE MtPareto(Arg1,Arg2) BEGIN
        TEMPORARY   Z, XI;
        STEP1: Z = Call("mtrand.dll", "?nextNum@@YAHXZ");
        IF (Z = 0)THEN GOTO STEP1;
        Z = Z/2147483647;
        XI = Arg1#Z^(-1/Arg2);
        RETURN (XI);
        END;
```

Рис. 7. Модель исследуемой СМО Pa/Pa/1/*m* при *m* = 25

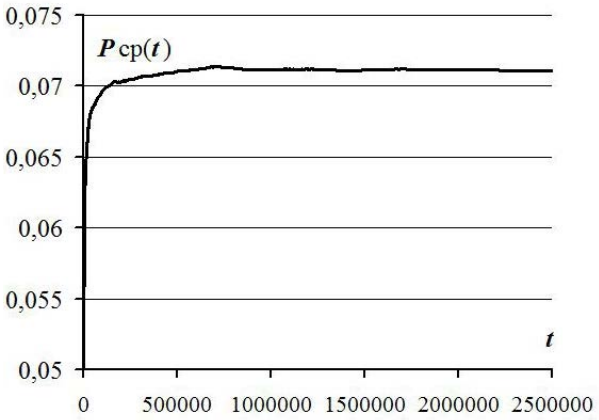

Рис. 8. ПП $P_{ср}(t)$ при *m* = 25

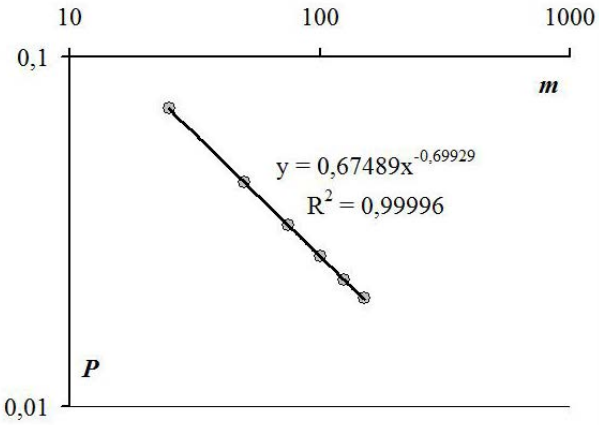

Рис. 9. Найденная путем ИМ степенная зависимость *P(m)*

больше максимально допустимого числа обращений к стандартному ГСЧ Uniform (равному длине его периода). Таким образом, при использовании Uniform подобное усреднение можно было бы проводить лишь по вдесятеро меньшему числу реализаций ПП, получая при этом в $10^{0,5} \approx 3$ раза большие погрешности (большую ширину доверительной полосы).

Заметим, что использование МТ-генератора, имеющего длину периода около 10^{6000}, в принципе позволяет увеличить число прогонов в этом эксперименте до (10^{6000}/20 000 000 000) раз, уменьшая за счет этого статистические погрешности получаемых оценок приблизительно в 10^{3000} раз, т.е. обеспечивая около 3 тысяч точных десятичных цифр после запятой. Конечно, такое увеличение числа прогонов практически невозможно из-за пропорционального ему увеличения времени ИМ. Но и оценки с 3 тысячами точных цифр в ИМ на практике не нужны. Поэтому МТ-генератор при ИМ на GPSS просто обеспечивает практически неограниченные возможности повышения точности за счет увеличения объемов выборки, снимая накладываемые генератором Uniform на длину выборок ограничения, ставшие в настоящее время критическими в связи с быстро растущим быстродействием компьютеров [3, 8].

6. Пример моделирования СМО с РТХ с применением МТ-генератора. В качестве одной из основных задач моделирования СМО с РТХ в [8] обосновывается задача определения размера *m* буфера, достаточного для того, чтобы стационарная вероятность *P* потери заявки не превышала заданной величины. Покажем, как с применением МТ-генератора решается эта задача при ИМ системы Pa/Pa/1/*m* (где символы Pa соответствуют распределению Парето с порогом *K* и параметром формы α). Параметры распределений Парето заданы, соответственно, в виде $K_1 = 1$, $\alpha_1 = 1{,}25$ (для входящего потока) и $K_2 = 1$, $\alpha_2 = 1{,}5$ (для времени обслуживания).

Модель этой СМО при размере буфера $m = 25$ представлена на рис. 7. Процедура MtPareto(K,α), реализующая РТХ Парето и вызываемая из модели, транслируется вместе с использующей ее программой. Выполнен 10001 независимый прогон модели, каждый на интервале модельного времени (0; 2 500 000). При этом произошло около 10 млрд обращений к МТ-генератору «Вихрь Мерсенна». Соответствующая 10001 траектория изменения во времени оценки вероятности отказа (в виде доли потерянных заявок от всех пришедших) накоплена в массиве и перед завершением моделирования усреднена.

Полученная усредненная траектория $P_{ср}(t)$ изображена на рис. 8. В середине моделируемого интервала времени ПП практически заканчивается. Усредняя во второй половине этого интервала все значения $P_{ср}(t)$ по времени *t*, получаем оценку для вероятности *P* при $m = 25$, составляющую 0,07114.

Аналогичные эксперименты при $m = 50$, 75, 100, 125 и 159 дают для *P* оценки 0,0438, 0,0329, 0,0269, 0,0230 и 0,0204 соответственно. На получение этих шести точек (m, P) зависимости $P(m)$ потрачено около суток компьютерного времени. Все шесть точек идеально легли в координатной плоскости с логарифмическими масштабами осей на прямую (рис. 9). Это подтверждает высокую точность полученных оценок для $P(m)$, поскольку эта зависимость при больших m является степенной [11]. Получаемая с высокой точностью линия тренда дает для $P(m)$ уравнение $P = 0{,}675m - 0{,}7$. Отсюда легко выводится искомая зависимость $m(P)$: $m = (P/0{,}675) - (1/0{,}7) \approx 0{,}57P - 1{,}43$, которую можно использовать для расчета требуемых m при любых $P \le 0{,}0438$.

Учитывая, сколь большое число обращений к ГСЧ потребовалось в описанном эксперименте, нетрудно заметить, что если бы в нем использовались стандартные ГСЧ, встроенные в GPSS World, то такая точность при моделировании не могла бы быть достигнута. Ознакомление с прочими преимуществами МТ-генератора только укрепляет этот вывод [4]. На практике он подтверждается и большим количеством неудач, постигших автора статьи при использовании стандартных ГСЧ GPSS World

при ИМ СМО с РТХ. Ни в одном таком опыте статистические оценки зависимостей *P(m)* не ложились столь очевидным образом на прямую, как это происходит в опытах с применением генератора MtRand (рис. 9).

7. Выводы. Ограниченные возможности генераторов случайных чисел языка GPSS World не позволяют в полной мере реализовать быстродействие современных компьютеров для повышения точности результатов имитационного моделирования. Эту проблему можно решить применением внешних генераторов.

Хорошим выходом является применение MT-генератора с длиной периода ($2^{19\,937}$–1), основанного на алгоритме «Вихрь Мерсенна».

Изложенные в статье подходы и опыт применения MT-генератора показывают, что этот генератор позволяет качественно повысить точность и надежность результатов моделирования СМО на GPSS World, в том числе таких СМО, которые описываются распределениями с тяжелыми хвостами.

Библиографический список

1. Руководство пользователя по GPSS World / Под ред. К. В. Кудашова ; пер. с англ. – Казань : Мастер Лайн, 2002. – 384 с.

2. Задорожный, В. Н. Реализация больших выборок при моделировании систем массового обслуживания на GPSS World / В. Н. Задорожный // ИММОД-2015 : материалы тр. VII Всерос. науч.-практ. конф. В 2 т. Т. 1. – М., 2015. – С. 225–229.

3. Задорожный, В. Н. Особенности моделирования систем массового обслуживания с тяжелыми хвостами распределений на GPSS World. Метод ARAND / В. Н. Задорожный // Омский научный вестник. Сер. Приборы, машины и технологии. – 2015. – № 3 (143). – С. 307–311.

4. Вихрь Мерсенна [Электронный ресурс]. – Режим доступа : http://dic.academic.ru/dic.nsf/ruwiki/88739 (дата обращения: 20.12.2015).

5. Задорожный, В. Н. Проблемы генерации случайных величин с фрактальными распределениями / В. Н. Задорожный, О. И. Кутузов // Омский научный вестник. Сер. Приборы, машины и технологии. – 2012. – № 3 (113) – С. 20 24.

6. Zadorozhnyi, V. N. Cascade Method of Realization of Heavy-Tailed Distributions in Data Network Modelling. 2015 International Siberian conference on control and communications SIBCON, sec. Control of the Large-Scale Systems, Russia, Omsk, May 21–23, 2015.

7. Задорожный, В. Н. Каскадный метод реализации распределений с тяжелыми хвостами / В. Н. Задорожный // Омский научный вестник Сер. Приборы, машины и технологии. – 2015. – № 140. – С. 222–226.

8. Zadorozhnyi V. N. Fractal Queues Simulation Pecularities / Communication in Computer and Information Science, Vol. 564 : Information Technologies and Mathematical Modelling, 14th International Scientific Conference, ITMM 2015.

9. Харин, Ю. С. Практикум на ЭВМ по математической статистике / Ю. С. Харин, М. Д. Степанова. – Мн., 1987. – 304 с.

10. Клейнрок, Л. Вычислительные системы с очередями / Л. Клейнрок ; пер. с англ. Б. С. Цыбакова. – М. : Мир, – 1979. – 600 с.

11. Zadorozhnyi V. N. Simulation modeling of fractal queues / Dynamics of Systems, Mechanisms and Machines (Dynamics), 2014 DOI: 10.1109/Dynamics.2014.7005703 Publication Year: 2014, P. 1–4.

ЗАДОРОЖНЫЙ Владимир Николаевич, доктор технических наук, доцент (Россия), профессор кафедры «Автоматизированные системы обработки информации и управления».
Адрес для переписки: zwn2015@yandex.ru

Статья поступила в редакцию 21.12.2015 г.

Книжная полка

621.39/В27

Величко, В. В. Модели и методы повышения живучести современных систем связи / В. В. Величко, Г. В. Попков, В. К. Попков. – М. : Горячая линия-Телеком, 2014. – 269 с. – ISBN 978-5-9912-0408-8.

Рассмотрены вопросы анализа живучести сетей связи в условиях разрушающих информационных воздействий. Дана классификация информационных атак в информационных сетях и методы их обнаружения. Уделено значительное внимание вопросам, связанным с живучестью и надёжностью мобильных систем связи, предложены модели структурной надежности.

004.4/В14

Вайсфельд, М. Объектно-ориентированное мышление / М. Вайсфельд ; пер. с англ. В. Черник. – СПб. : Питер, 2014. – 303 с. – ISBN 978-5-496-00793-1.

Объектно-ориентированное программирование — это фундамент современных языков программирования, включая C++, Java, C#, Visual Basic, .NET, Ruby и Objective-C. Кроме того, объекты лежат в основе многих веб-технологий, например JavaScript, Python и PHP. Объектно-ориентированное программирование обеспечивает правильные методики проектирования, переносимость кода и его повторное использование, однако для того, чтобы все это полностью понять, необходимо изменить свое мышление. Разработчики, являющиеся новичками в сфере объектно-ориентированного программирования, не должны поддаваться искушению перейти непосредственно к конкретному языку программирования (например, Objective-C, VB .NET, C++, C#, .NET или Java) или моделирования (например, UML), а вместо этого сначала уделить время освоению того, что автор книги Мэтт Вайсфельд называет объектно-ориентированным мышлением. Несмотря на то, что технологии программирования изменяются и эволюционируют с годами, объектно-ориентированные концепции остаются прежними — при этом неважно, какой именно является платформа.