Лабораторна робота № 2
Дискретно-подієвого моделювання.
Моделювання систем масового обслуговування

Задача. Змоделювати роботу квиткових кас. У каси є єдина черга, яку обслуговують дві основні каси. Якщо основні каси не справляються з потоком покупців, то відкривається третя каса.

Потік покупців змінюється в залежності від часу доби і стає більшою у вихідні дні. Розклад потоку покупців наведений нижче.

Робочі дні:

8: 00-13: 00 – десять осіб на годину;

13: 00-16: 00 – п'ятнадцять осіб на годину;

16: 00-22: 00 – двадцять осіб на годину.

Вихідні:

9: 00-12: 00 – двадцять осіб на годину;

12: 00-21: 00 - сорок осіб на годину.

Покупці, час очікування покупки у яких перевищила годину, йдуть з кас, не купивши квитка. Час обслуговування одного покупця в касах змінюється випадковим чином від 2 до 15 хвилин і в середньому становить 5 хвилин. Передбачити в моделі облік купили і не купили квитки.

Розв'язання

Етап 1. Задання логіки роботи моделі

Крок 1. *Створення нової моделі*

Створіть нову модель (рис. 1) в середовищі AnyLogic, натиснувши на кнопку *Створити* і вибравши *Модель* з меню, що випадає (або використати клавіші швидкого доступу <*Ctrl + N*>).

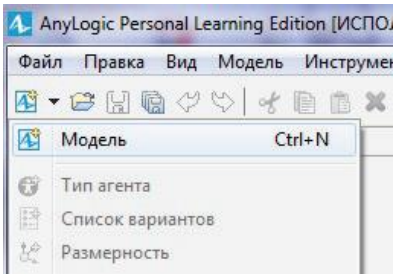


Рис. 1. Створення нової моделі

Введіть ім'я моделі *ticket* в діалоговому вікні та задайте одиниці модельного часу – хвилини (рис. 2). Натисніть кнопку *Готово*.

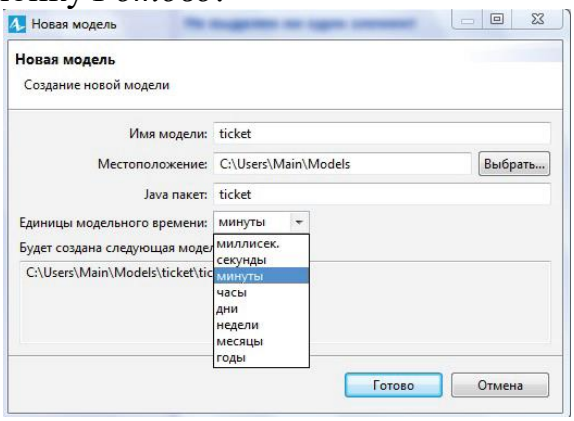


Рис. 2. Завдання параметрів нової моделі

Крок 2. *Моделювання приходу покупців*

У вікні моделі перейдіть на вкладку *Палітра* і відкрийте першу бібліотеку зі списку – *Бібліотеку моделювання процесів*. З неї перетягніть на робоче поле блок *Source* (рис. 3).

Як відомо, саме цей блок моделює поява заявок у моделі. У нашому випадку покупці – це заявки.

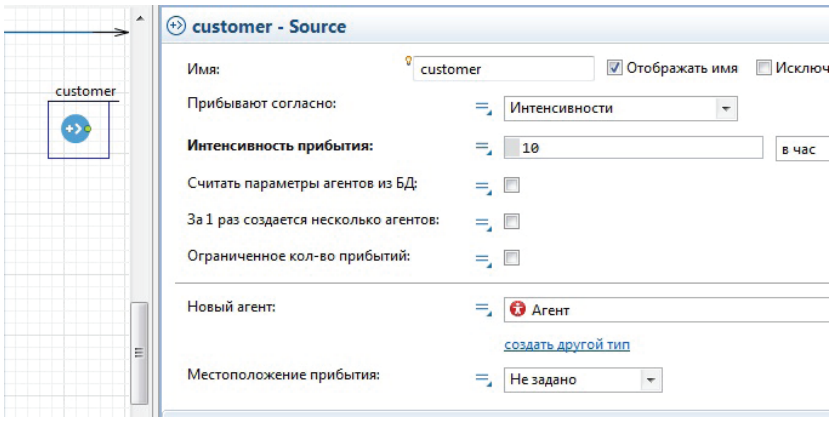


Рис. 3. Моделювання приходу покупців

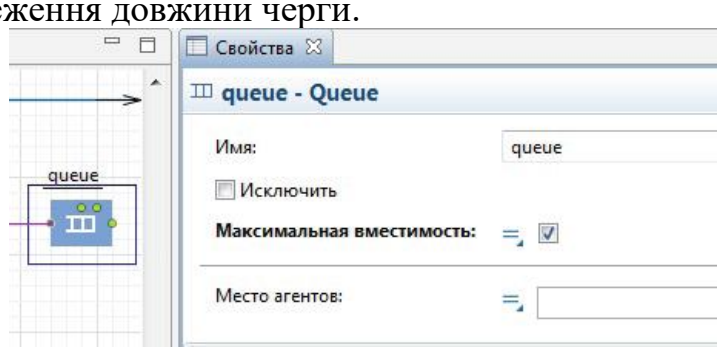
На початку моделювання ми не будемо враховувати зміну інтенсивності потоку покупців протягом дня і тижня. просто задамо найменшу межу потоку. Для цього виберемо в пункті *Прибувають згідно* режим *Інтенсивності* і задамо значення інтенсивності 10. Це означає, що в середньому 10 разів на годину будуть приходити покупець, тобто приблизно кожні 6 хвилин.

Крок 3. *Моделювання черги покупців*

Перетягніть з бібліотеки моделювання процесів блок *Queue*.

Будьте уважні і постарайтеся зробити так, щоб блок *Queue* пов'язався з блоком *Source*. Це дозволить уникнути зайвого клопоту по домальовані зв'язків між блоками. Якщо не вдасться пов'язати блоки, то потрібно двічі клацнути на вихідний порт блоку *Source* і потягнути зв'язок до блоку *Queue*. Заявка проходить в моделі тільки по зв'язкам. Якщо блоки не пов'язані, то заявка не зможе перейти в наступний блок.

Блок *Queue* імітує накопичення заявок в моделі, фактично моделює чергу. Залишимо всі її параметри за замовчуванням. Це означає, що її логіка буде відповідати логіці черги «перший прийшов, перший пішов »і не буде задано ніякої прив'язки на місцевості. Відмітити пункт Максимальна місткість (рис. 4), оскільки в завданні нічого не було сказано про обмеження довжини черги.



Мал. 1.4. Моделювання черги покупців

Крок 4. *Моделювання процесу покупки квитків*

Для процесу покупки квитків потрібно ресурс – касири. Для моделювання ресурсів у моделі використовується блок *ResourcePool*.

Перетягніть його на робоче поле моделі. Цей блок не зв'язується з жодним блоком в моделі, оскільки через нього не повинні проходить заявки.

У властивостях блоку задайте його ім’я booking_clerk. Оскільки касири можуть пересуватися від каси до каси, то задайте тип ресурсу як Рухомий і вкажіть їх швидкість 2 км / год. Поки не задаватимемо розклад роботи касирів і в пункті *Кількість задано* залишимо режим *Напряму* і вкажемо кількість касирів, наприклад 2 (рис. 5).

Перед тим як моделювати сам процес покупки квитків, використовуємо блок вибору руху заявок, оскільки покупці з черги можуть йти як в першу, так і в другу касу. перетягніть блок *selectOutput* на робоче поле і з’єднайте його з блоком *Queue*. Умова вибору кас введемо пізніше.

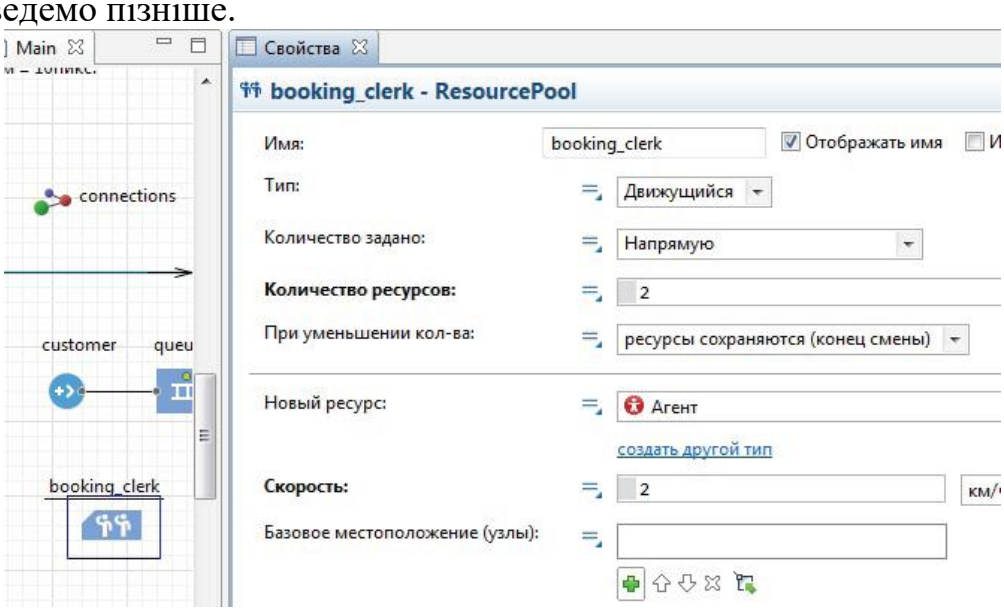


Рис. 5. Моделювання ресурсів для продажу квитків

Далі потрібно змоделювати сам процес покупки квитків. Процес покупки квитків – це процес обслуговування заявки. Для цього використовується блок *Service*, який включає в себе логіку роботи трьох блоків *Sieze* (захоплення ресурсів), *Delay* (утримання заявки і ресурсів), *Release* (звільнення ресурсів). Також блок *Service* має власну чергу, тобто в нього ще вбудований блок *Queue*. Тому якби до кожної касі йшла своя чергу, то блок *Queue* окремо можна було б не використати.

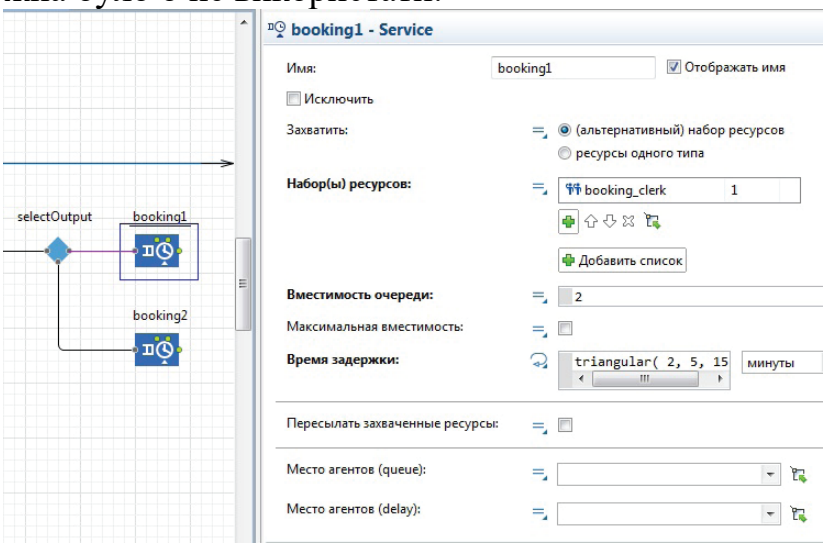


Рис. 6. Моделювання процесу продажу білетів

Перетягніть два блоки *Service* на робоче поле. У властивостях блоків задайте їх імена booking1 і booking2. Задайте місткість власних черг 2 і час затримки (рис. 6). Час затримки задається функцією трикутного розподілу із середнім значенням 5 хвилин, мінімальним значенням 2 хвилини і максимальним значенням 15 хвилин. У пункті Набір

(и) ресурсів виберіть зі списку ресурсів щойно створений ресурс booking_clerk. Тепер в першій і в другій касах будуть працювати зазначені в ресурсах 2 касира.

Тепер, коли обидві каси промоделювати, повернемося до умови вибору каси покупцем. Нехай покупець йде в другу касу, побачивши, що в першій касі зібралося більше трьох осіб. Така поведінка заявкам можна задати, якщо використовувати в умови вибору функцію об'єкта service – size (), яка повертає кількість клієнтів, що обслуговуються в даний момент об'єктом. Якщо величина, повернена функцією size () об'єкта booking1, буде більше 3 (рис. 7), то означає, що в цю касу покупець не піде.

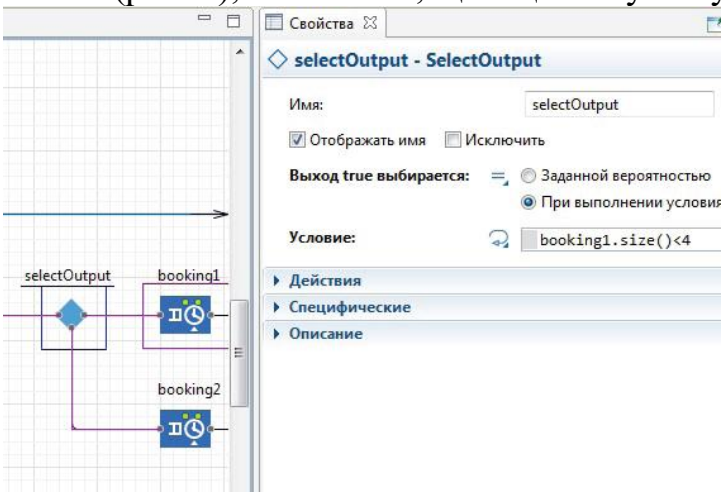


Рис. 7. Умова вибору кас

Крок 5. *Моделювання догляду покупців з кас*

Покупці, купивши квиток, йдуть з кас. Для моделювання цієї події використовується об'єкт Sink (рис. 8). Перетягніть об'єкт Sink на робоче поле і з'єднайте його з виходами обох кас. У властивостях об'єкта задайте його ім'я exit.

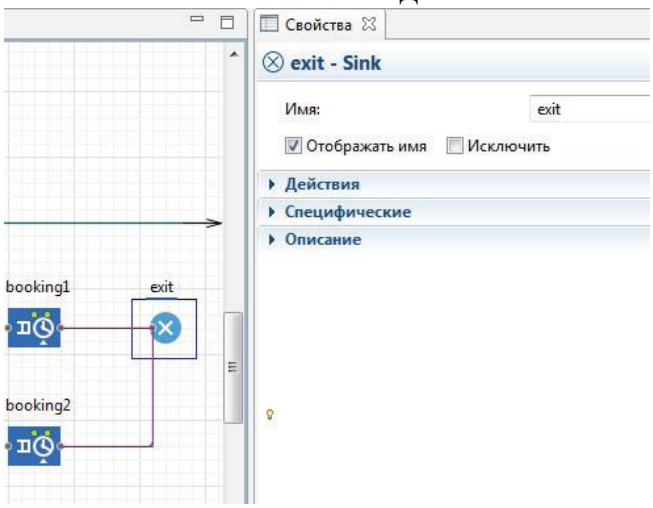


Рис. 8. Моделювання, як покупець залишає касу

Крок 6. *Перевірка працездатності моделі*

Малюнок моделі закінчений. Він повинен виглядати, як на рис. 9.

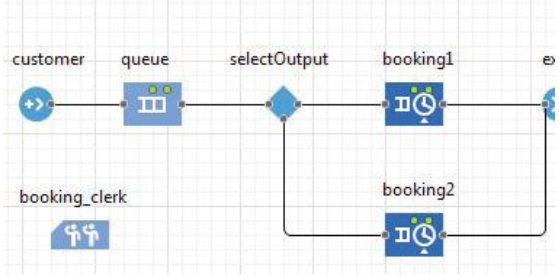


Рис. 9. Чернетка моделі

Запустіть модель. Результат роботи моделі через деякий модельний час повинен виглядати, як на рис. 10.

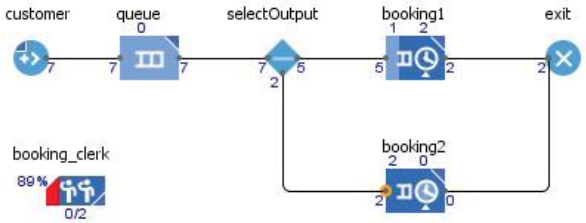


Рис. 10. Робота моделі

Етап 2. *Задання розкладів*

Крок 1. *Задання розкладу приходу покупців*

У завданні було сказано, що інтенсивність приходу покупців змінюється протягом дня і стає більша у вихідні. Будь-які періодичні зміни в моделі зручно задавати об'єктом *Розклад*. Перетягніть об'єкт *Розклад* з бібліотеки на робоче поле моделі. Цей об'єкт не потрібно об'єднувати ні з яким об'єктом.

У властивостях об'єкта задайте його ім'я *schedule_customer*. Оскільки буде здаватися розкладом *Зміна інтенсивності прибуття покупців*, то вибираємо тип розкладу – *Розклад інтенсивності*.

Тип розкладу задається в розділі *Дані*. Значення за замовчуванням залишаємо 0, це буде означати, що в години, не зазначені в розкладі, покупці не приходять. Розклад буде задавати інтервали, в яких інтенсивність залишається постійною. Задаємо розклад на тиждень, тому тривалість розкладу вибираємо *Тиждень*. Саме розклад задається в таблиці в пункті *Повторювати розклад щотижня*. Праворуч від таблиці є кнопки для додавання (+), видалення (x) і прокрутки розкладу. Натисніть на кнопку «+» для додавання нового рядка в таблицю. Кожен рядок в таблиці розкладу задає один часовий інтервал, початок і кінець якого задаються у заголовках таблиці. У стовпці *Значення* задайте значення інтенсивності прибуття покупців. В результаті має вийти розклад, як на рис. 11.

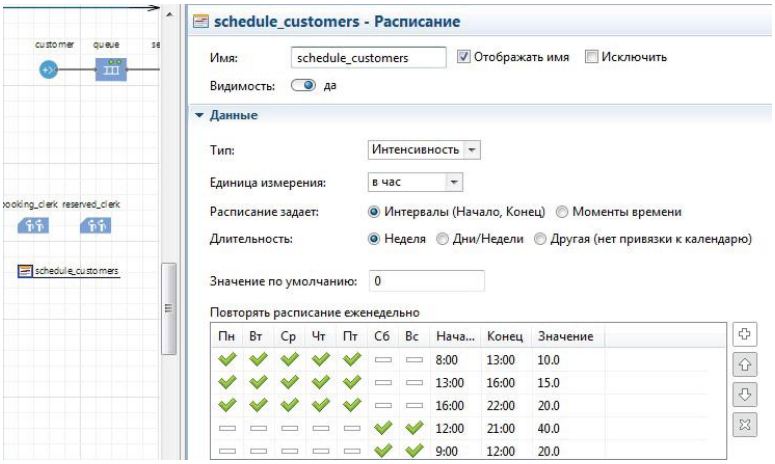


Рис. 11. Розклад інтенсивності прибуття покупців

Крок 2. *Задання розкладу роботи касирів*

Розклад роботи касирів буде задавати час роботи касирів і кількість працюючих касирів (рис. 12). Оскільки кількість касирів – це ціле число, то в пункті *Тип* вибираємо ціле. *Значення за замовчуванням* задаємо 0, значення протягом робочого часу – 3.

Крок 3. *Задання розкладу перерв у роботі касирів*

В роботі касирів передбачаються 15-хвилинні перерви, під час яких касири як ресурси недоступні. Отже, для задання розкладу перерв потрібно створити розклад доступності, для якого вибираємо Тип: так / ні (рис. 13). За замовчуванням вказуємо ресурс доступним, тобто значення так. У стовпці перерв указувати значення немає. Також у властивостях розкладу задаємо його ім'я schedule_timeout.

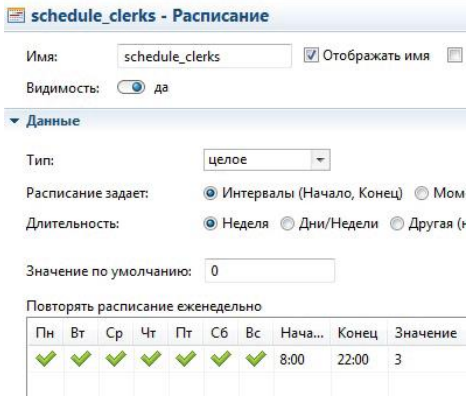


Рис. 12. Розклад роботи касирів

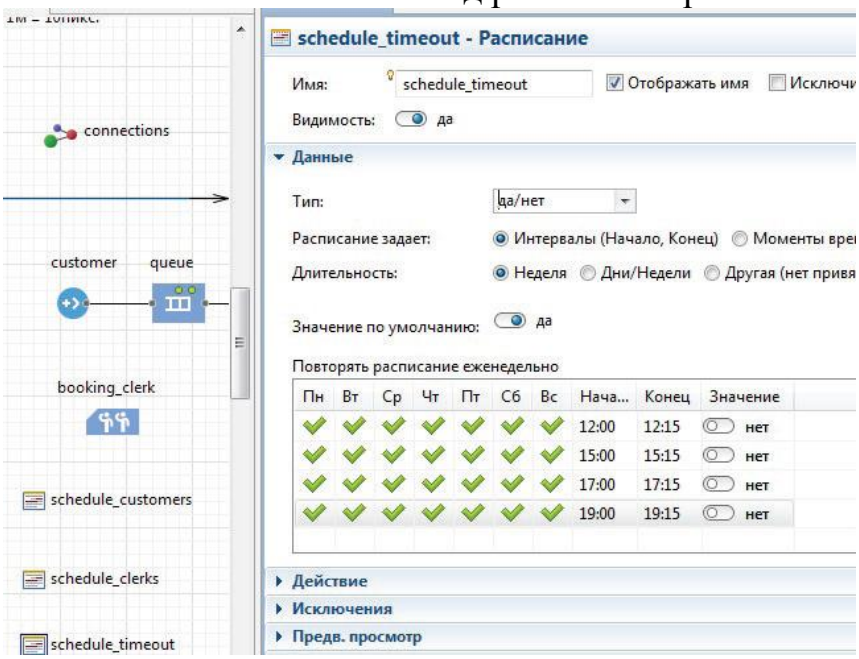


Рис. 13. Розклад перерв роботи касирів

Крок 4. *Прив'язка розкладу приходу покупців до об'єкта source*

Спочатку створювали об'єкт *Source* і вказували, що заявки прибувають згідно інтенсивності, а потім ставили інтенсивність прибуття. Тепер прийшла пора виправити ситуацію. У властивостях об'єктивним та *source* в пункті Прибувають згідно виберіть варіант розкладу інтенсивностей (рис. 14). З'явиться новий пункт у властивостях Розклад інтенсивностей. Виберіть в цьому пункті створений раніше розклад прибуття покупців.

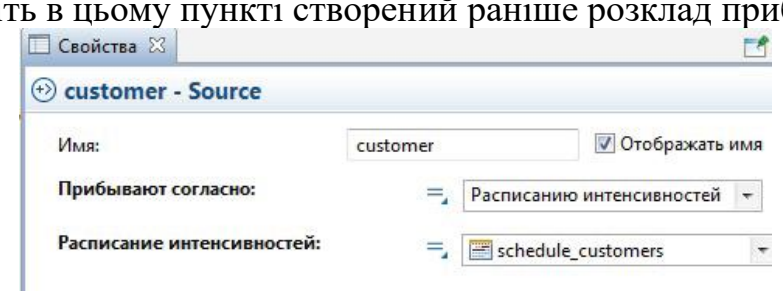


Рис. 14. Прив'язка розкладу прибуття покупців

Тепер покупці будуть приходити з різною інтенсивністю протягом дня.

Крок 5. *Прив'язка розкладу роботи касирів до об'єкта ResourcePool*

Раніше нами кількість ресурсів в об’єкті booking_clerk було задано безпосередньо. Тепер задамо розклад їх роботи (рис. 15). У пункті *Кількість задано* виберіть варіант *Розкладом* і в пункті *Розклад* який з’явився виберіть раніше створене розклад роботи касирів.

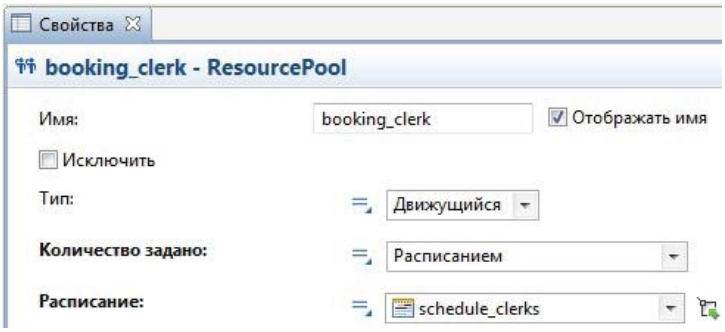


Рис. 15. Прив’язка розкладу роботи касирів

Крок 6. *Прив’язка розкладу перерв в роботі касирів*

Відкрийте розділ *Зміни, перерви, аварії, обслуговування ...* у властивостях об’єкта *booking_clerk* (рис. 16). Поставте прапорець в пункті *Перерви*. З’явиться вікно, в якому потрібно задати розклад перерв. Виберіть раніше створений розклад перерв роботи касирів у пункті *Розклад перерв*.

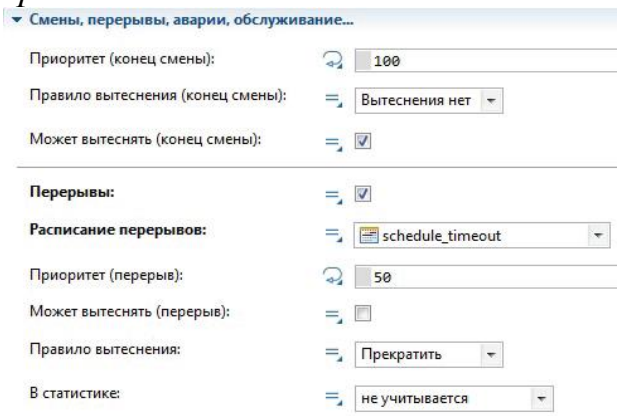


Рис. 16. Прив’язка розкладу перерв у роботі касирів

Етап 3. *Створення резервної каси і втрати клієнтів, чиї очікування покупки перевищило годину*

Крок 1. *Моделювання резервної каси*

Для моделювання резервної каси використовується блок *service* (Рис. 17) і всі властивості задаються такі ж, як і у перших двох касах. Тільки час обробки задається трохи менший, ніж в перших двох касах.

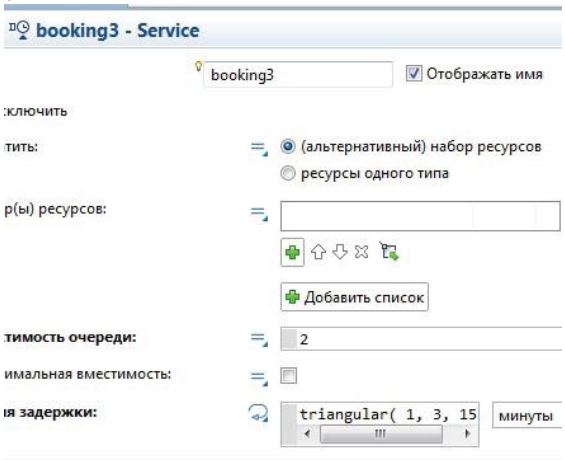


Рис. 17. Моделювання резервної каси

Крок 2. *Завдання ресурсів для резервної каси*

У резервній касі також працюватимуть касири, але з іншої бригади, тому потрібно створити новий ресурс. Назвемо його reserved_clerk (Рис. 18). Розклад роботи касирів резерву такий самий, як і у основної бригади.

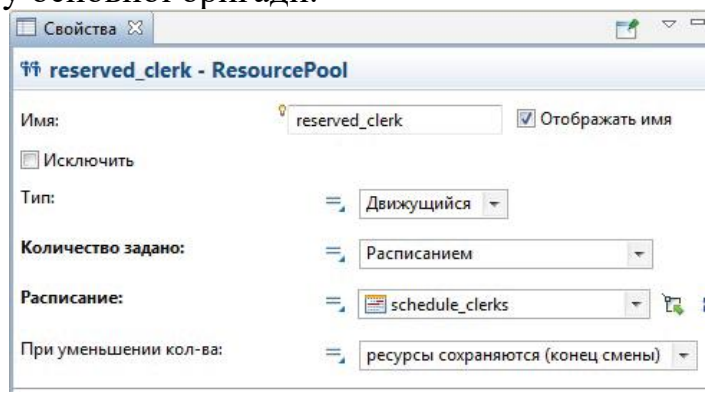


Рис. 18. Моделювання резервної бригади касирів

Крок 3. *Прив'язка ресурсів до резервної каси*

Для того щоб прив'язати резервну бригаду до резервної каси, вкажіть в пункті *Набір (и)* ресурсів щойно створений ресурс reserved_clerk (рис. 19). Вихід каси з'єднайте з об'єктом *exit*.

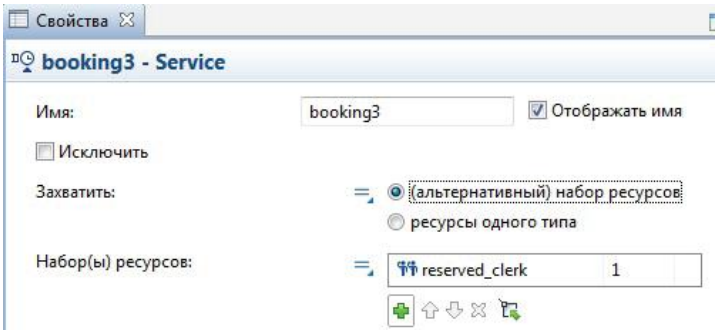


Рис. 19. Прив'язка ресурсів до резервної касі

Крок 4. *Організація обмеження часу очікування в касах*

В умові задачі було сказано, що клієнти, чиє очікування покупки квитка перевищила годину, йдуть з кас, не купивши квитки. Для реалізації цього потрібно задати тайм-аут, після якого заявка йде з об'єкта service через вихід OutTimeout. Час тайм-ауту задається в розділі *Специфічні* (рис. 20). Задайте тайм-аути в першій і другій касі, рівні 30 хвилин, в третій касі – 20 хвилин.

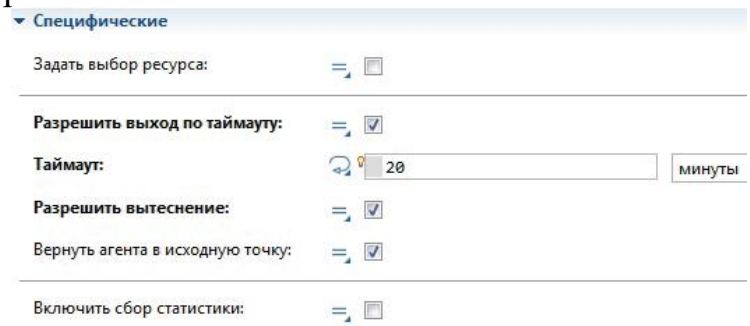


Рис. 20. Завдання тайм-ауту

Крок 5. *Організація виходу заявок з першої і другої каси, чиє час очікування перевищило 30 хвилин*

З'єднайте виходи OutTimeout об'єктів booking1 і booking2 зі входом в об'єкт booking3. Таким чином, покупці, чий час очікування в першій і другій касах перевищило 30 хвилин, йдуть в резервну касу.

У підсумку модель повинна вийти, як показано на рис. 21.

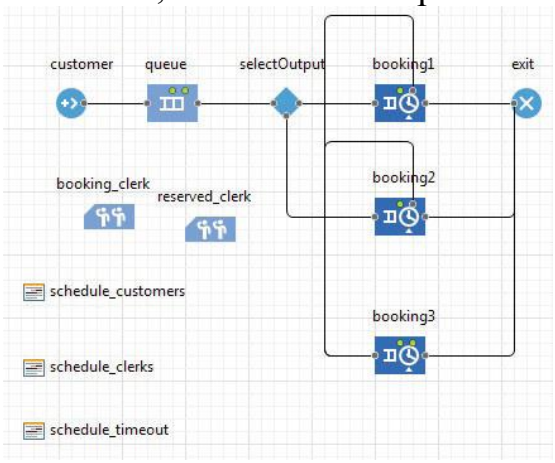


Рис. 21. Модель з урахуванням резервної каси

Крок 6. *Перевірка працездатності моделі*

Запустіть модель. Через деякий модельне час має отриматися, як показано на рис. 22.

Крок 7. *Додавання об'єкта Годинники*

Для зручності перегляду роботи моделі потрібно помістити на робоче поле об'єкт Годинники, які будуть показувати поточний модельний час (рис. 23). Цей об'єкт знаходиться на вкладці *Зображення* в палітрі інструментів. Перетягніть його на робоче поле.

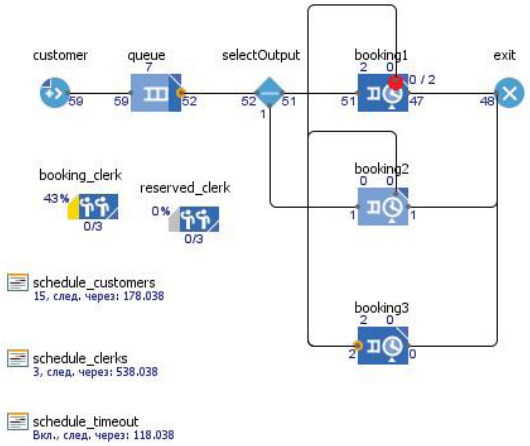


Рис. 22. Робота моделі з врахуванням резервної каси

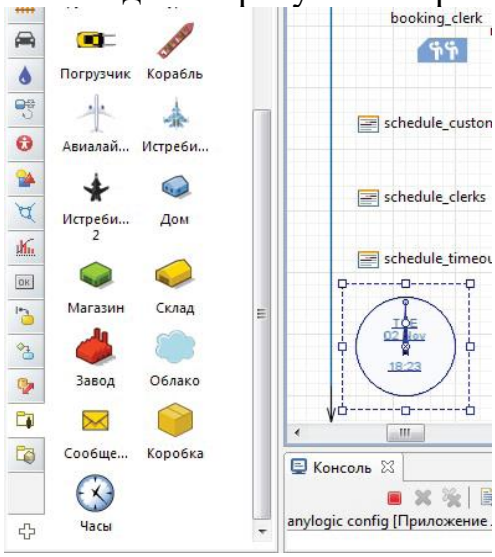


Рис. 23. Об'єкт Годинник

Крок 8. *Організація догляду покупців, що не купили квиток*

Відхід покупців, що не купили квиток, моделюється так само, як і догляд покупців, що купили квиток (рис. 24).

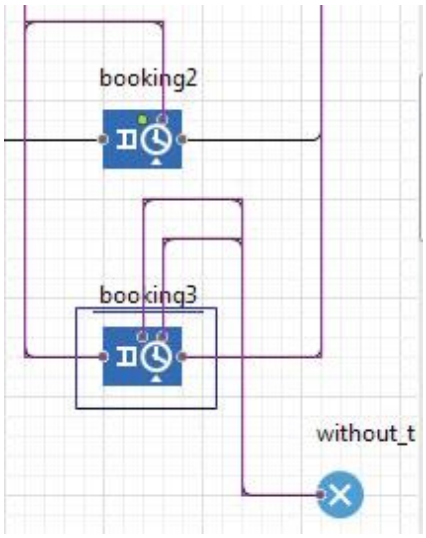


Рис. 24. Відхід покупців, що не купили квиток

Крок 9. *Перевірка працездатності моделі*

Запустіть модель. Має вийти, як на рис. 25.

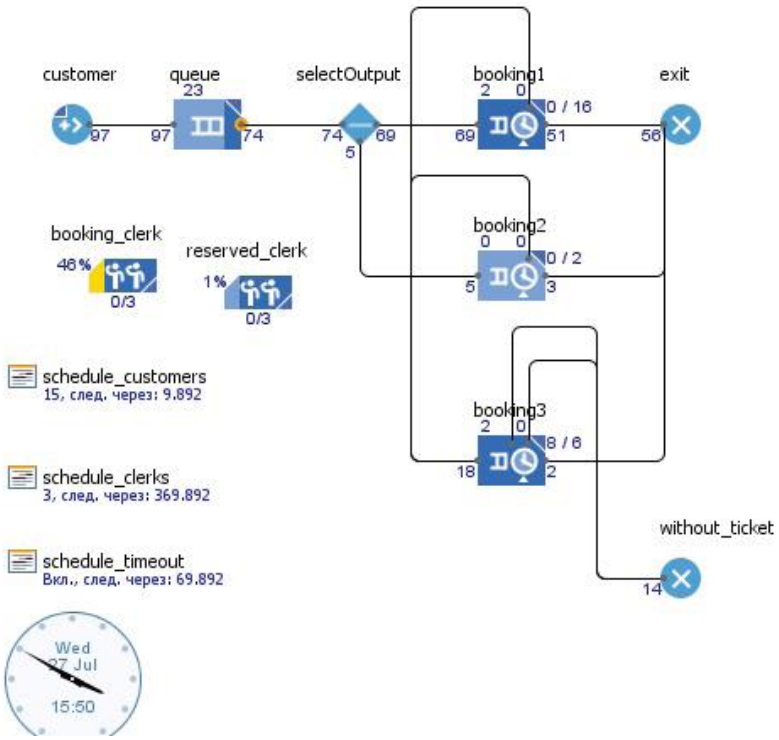


Рис. 25. Робота моделі з урахуванням відходу покупців, що не купили квиток

Етап 4. *Додавання динамічних графіків у модель*

Крок 1. Створення набору даних за проданими квитками

Будемо вважати, що кожен покупець купував 1 квиток. Звичайно, це трохи спрощений підхід. Для збору даних за кількістю проданих квитків у час роботи моделі використовується об’єкт *Набір даних* (рис. 26). Цей об’єкт – двовимірний масив, що зберігає значення X і Y. Цей об’єкт знаходиться в бібліотеці *Статистика* на панелі інструмент. Перетягніть цей об’єкт на робоче поле. Перейдіть в його властивості. Задайте ім’я набору – sold_tickets. Оскільки нас цікавить динаміка продажу квитків, то по осі X поставимо модельний час.

По осі Y будемо зберігати кількість покупців, які пройшли через вихід exit. Цю величину повертає функція size () об'єкта Sink.

Дані в наборі будуть оновлюватися кожні три хвилини.

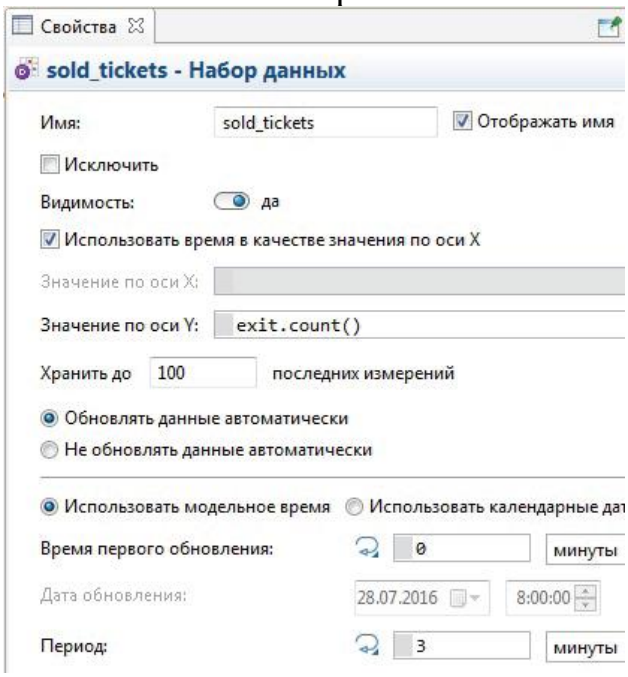


Рис. 26. Набір даних покупців, що купили квиток

Крок 2. *Створення набору даних за непроданими квитками*

Всі покупці, які пішли могли придбати мінімум один квиток. Будемо вважати мінімальні втрати. Для їх обліку створимо також набір даних *lost_tickets* за тією ж технологією, як і в кроці 1 (рис. 27).

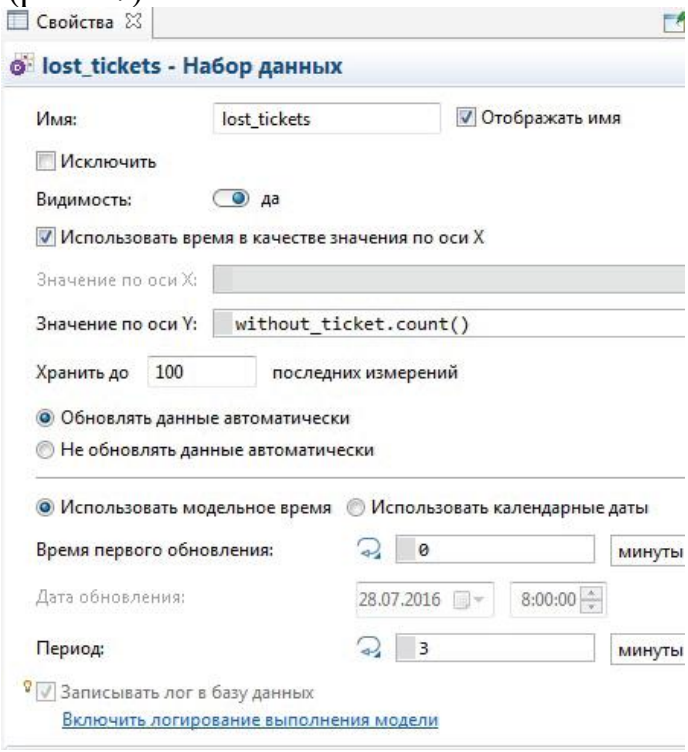


Рис. 27. Набір даних покупців, що не купили квиток

Крок 3. Створення кругової діаграми за покупцями

Для того щоб відстежувати в режимі роботи моделі співвідношення покупців, що купили і не купили квиток, використовуємо інструмент *Кругова діаграма* (рис. 28). Цей інструмент також знаходиться у бібліотеці *Статистика*. Перетягніть його на робоче

поле. Перейдіть у його властивості і у розділі *Дані* натисніть на «+», задайте назву для даних, а в поле *Значення* викличте метод *count ()* об'єкта *exit*.

Аналогічно задайте ще один набір даних в діаграмі.

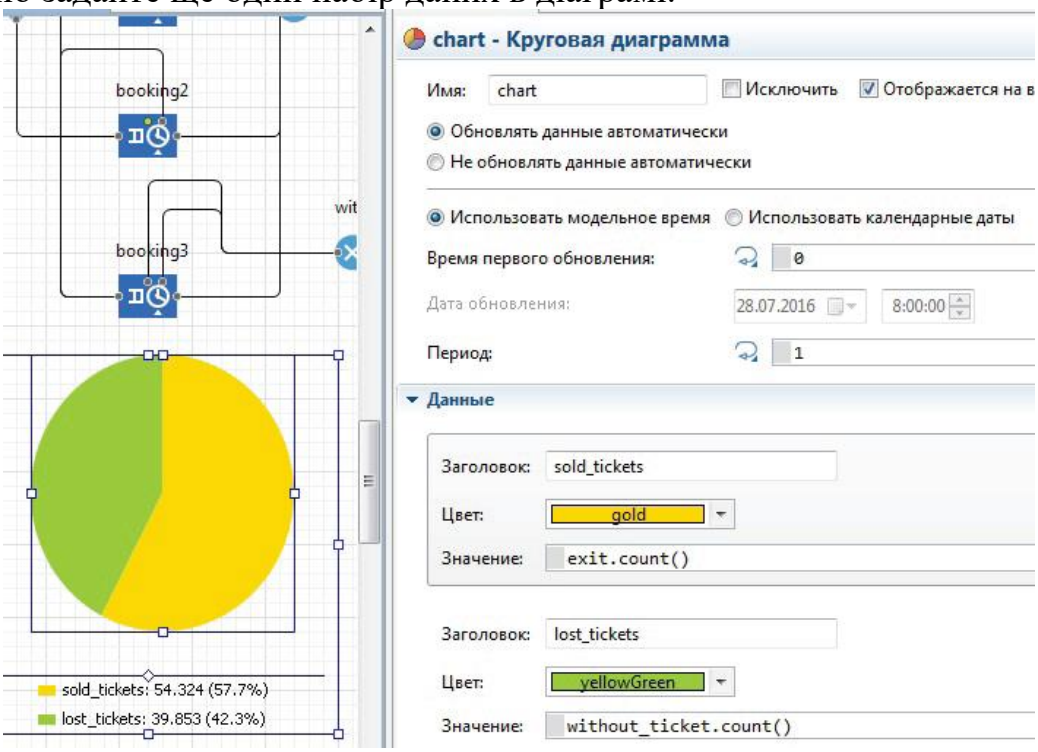


Рис. 28. Кругова діаграма

Крок 4. Створення тимчасового графіка, що відображає кількість покупців, що купили і не купили квиток

Щоб продемонструвати вміст наборів даних, створених на кроці 1 і кроці 2, використовуємо інструмент *Часовий графік* (рис. 29). Цей інструмент дозволяє відстежувати під час роботи моделі динамічно мінливі набори даних або значення. Інструмент також знаходиться на панелі *Статистика*. Перетягніть його на робоче поле і задайте в його властивостях створені раніше набори даних.

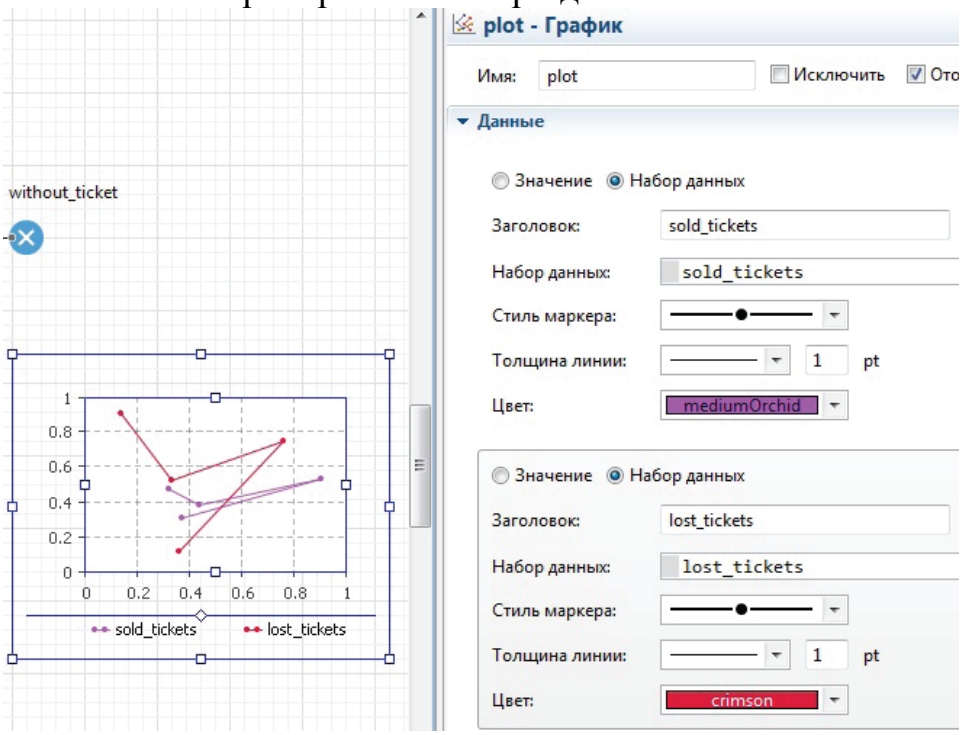


Рис. 29. Часовий графік

Крок 5. Перевірка працездатності моделі

Запустіть модель. Через деякий час роботи моделі має вийти, як на рис. 30.

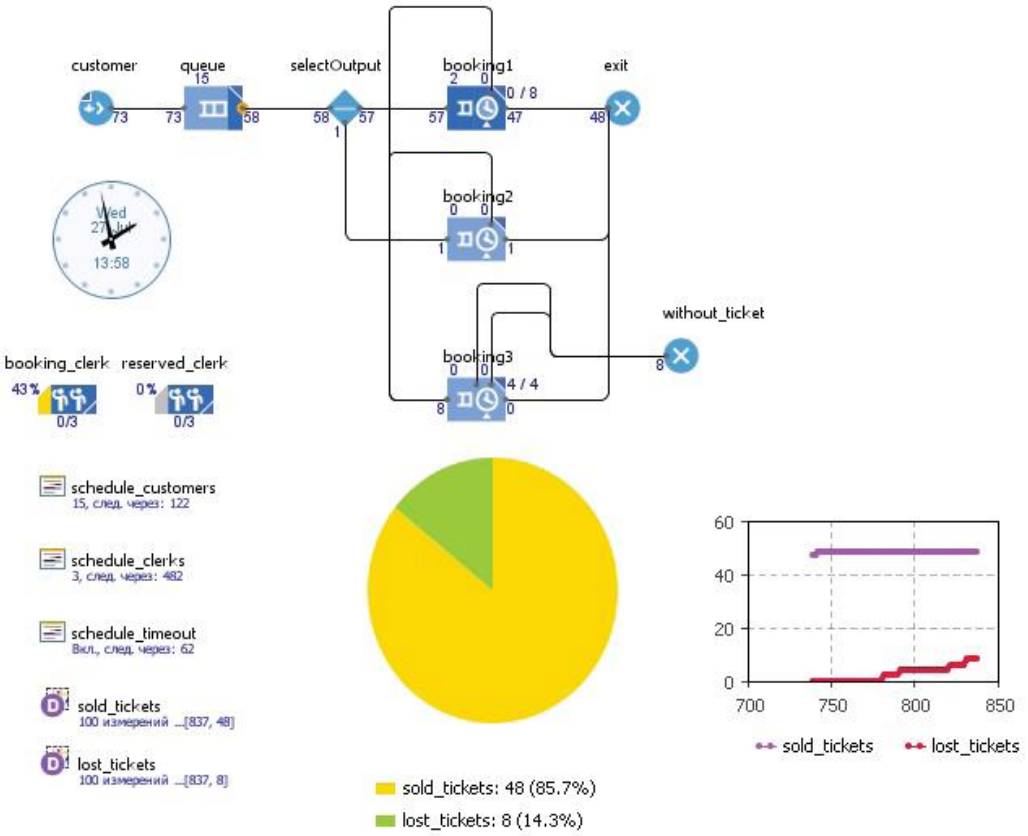


Рис. 30. Часовий графік

Самостійно

1. Виявити вузькі місця в моделі і запропонуйте рішення.
2. Творчий проект. Створіть модель масового обслуговування.