COS 214

2024

REPORT

CITY BUILDER

Milan Kruger
u04948123

Nicholas Dobson
u23671964

Sean van der Merwe
u22583387

Diyaana Jadwat
u23637252

Simon van der Merwe
u04576617

Amber Werner
u21457752

Introduction

Urban development and planning is the process of guiding and managing land use, urban environments, infrastructure, and related ecosystem and human services to promote optimal economic growth, enhance quality of life, ensure sustainable resource management, and support efficient infrastructure operation (Simon Elias Bibri, John Krogstie, 2017).

The aim of planning is to ensure urban areas are as efficient, attractive and functional as possible. For an urban area to thrive, a sound theoretical foundation that was planned and developed previously is a must. City management is a field that deals with how urban areas are developed and the constraints within which people behave. It deals with issues that emerge from urban development.

There are several components that work together to shape a city's main functionality and operations. These include transportation, buildings. citizens, government and taxes and utilities.

Citizens are the backbone of a city. Urban development should cater to citizens of that city to ensure maximum happiness and enhance life expectancy. Effective urban planning takes into account the needs of the residents in the city to run effectively.

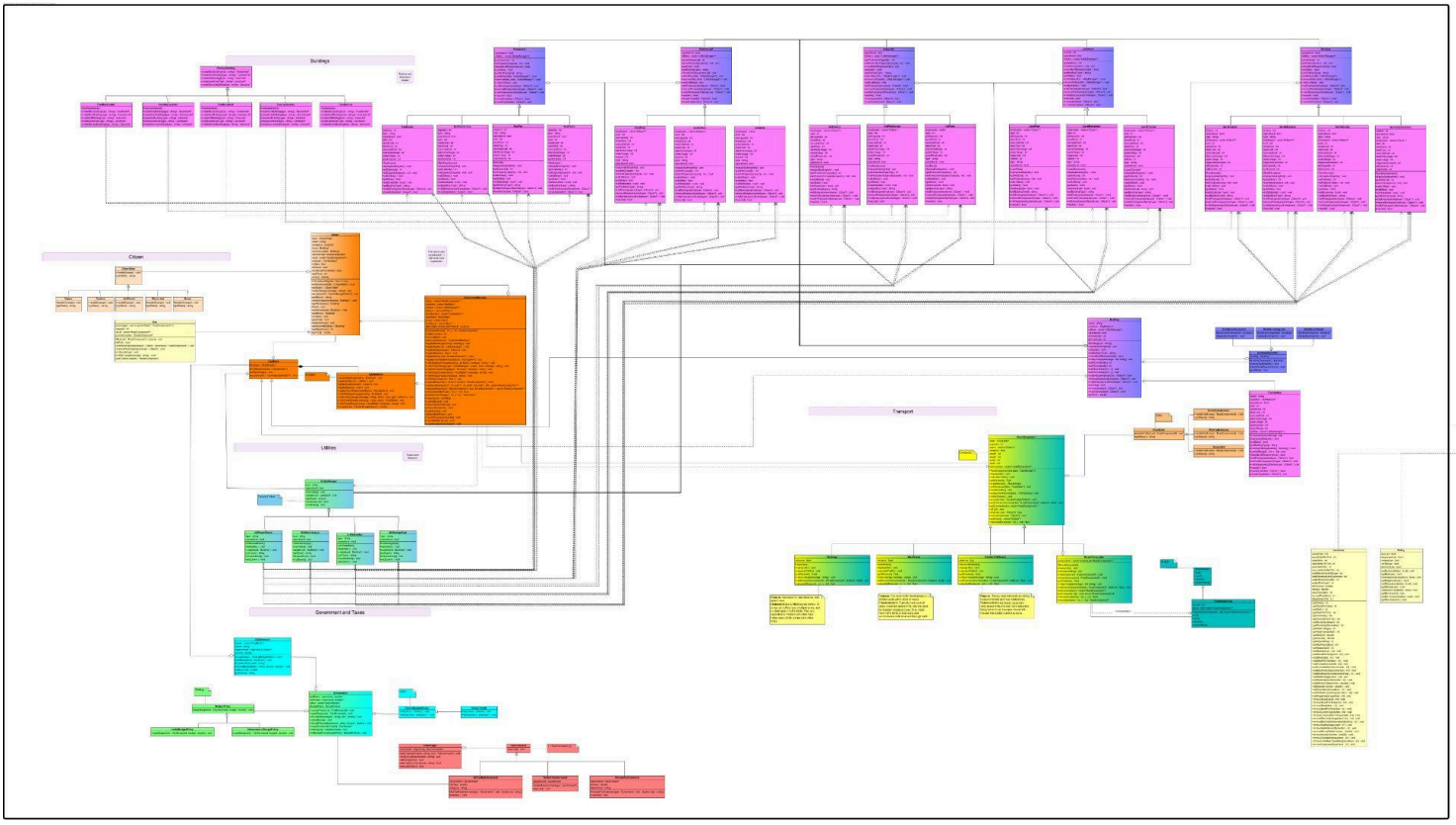
Buildings and services are the physical landscape of the city. They are needed for housing citizens, providing job opportunities, supplying resources to the city and enhancing all aspects of city living. Services are needed for a city to be liveable. If citizens are unhappy with services, they are likely to relocate or their quality of life may become poor, which can cause a decrease in life expectancy.

Transportation is vital in any city for mobility and connectivity. The layout of roads in a city is crucial to allow for minimum traffic, quick transport routes, tourism and accessibility to resources and jobs within the city.

A government is the backbone of a structured society. They implement laws and policies to make a city governable. They regulate services and ensure citizens are being provided with their rights to efficient services within the community. Taxes are necessary for funding operations within the city. They help keep the city functional and clean. With a government body in charge of taxes, they ensure that the money acquired is given back to the city to grow and develop.

Each of these components individually are important, but together they form an operational and thriving city with structure. Urban planning is needed to ensure that these components work together efficiently.

Applications of Design Patterns



Design Patterns Used in Project

Factory Method

Purpose:

In this simulation, the Factory Method centrally manages building instantiation by category (e.g., residential, industrial) and subcategory (e.g., houses, malls). Each main factory class, such as FactoryResidential or FactoryIndustrial, is responsible for its building type, providing:

- Modularity: Categories and subtypes are independently manageable.
- Ease of Maintenance: New buildings can be added by extending factories without modifying existing code.
- Scalability: Buildings can be customised without changing client code, allowing easy simulation expansion.

Why we used it:

The Factory Method pattern simplifies creating complex building objects (e.g., residential, commercial, industrial, landmark, services) without specifying exact classes. This approach

makes it easy to manage diverse building types with unique attributes, like citizen satisfaction or resource consumption, and supports future expansion when new building types are needed.

Participants:

Creator - FactoryBuilding

ConcreteCreator - FactResidential, FactCommercial, FactIndustrial, FactLandmarks, FactService

Product - Building, Residential, Commercial, Industrial, Landmark, Services

ConcreteProduct - ResHouse, ResTownhouse, ResFlat, ResEstate, ComShop, ComOffice, ComMall, IndFactory, IndWarehouse, IndPlant, LandPark, LandMonument, LandCCenter, ServHospital, ServEducation, ServSecurity, ServEntertainment

Used for creating all kinds of buildings needed for our simulation. This includes residential, commercial, industrial, landmarks and service buildings. This design pattern allows for the easy creation and expansion of the buildings in the city.

Decorator

Purpose:

The Decorator pattern enables flexible building "upgrades" that can improve citizen satisfaction or economic impact. Key benefits include:

- Modular Enhancements: Each feature is a separate decorator, easily added to buildings without cluttering base classes.
- Dynamic Functionality: Buildings can receive multiple improvements (e.g., repair, upgrade) at runtime without changing the core class.
- Maintainability and Flexibility: New features can be added as separate decorators, keeping code organised and adaptable.

Why we used it:

The Decorator pattern adds features (e.g., repair, economic boost, upgrade) to buildings dynamically without altering the core building classes. This keeps buildings focused on their main functions, avoiding rigid subclassing and making the system more flexible and extendable.

Participants:

Component - Building

ConcreteComponent - Residential, Commercial, Industrial, Landmark, Services

Decorator - BuildingDecorator

ConcreteDecorator - BuildingDecorEconomic, BuildDecorUpgrade, BuildDecorRepair

Permits upgrading or downgrading of buildings to either improve efficiency or reduce resource consumption. The Decorator design pattern has been applied to dynamically add additional functionalities to Building objects, without modifying the underlying structure. These functionalities include upgrades, repairs and economic features to improve the quality of buildings.

State

Purpose:

Allows the system to manage the transitions between states effectively based on citizen satisfaction or the road maintenance.

Why we used it:

It simplifies the management of citizens' reactions to city conditions.
It enhances the flexibility of how traffic is managed without modifying the road objects and components.

Participants for CitizenStates:

Context - Citizen

State - CitizenState

ConcreteState - Upset, Discontent, Indifferent, Content, Happy

Citizens have different emotional states – Happy, Content, Indifferent, Discontent, Upset.

Participants for Road States:

Context - RoadComponent

State - RoadState

ConcreteState - UnderConstruction, WorkingNoIssues, Congested

Our system requires roads that can impact traffic flow, thus roads have different physical states and way traffic is handled is affected by the state of the road.

Strategy

Purpose:

The government needs to be able to implement different policies.

Why we used it:

Policies are introduced by the Strategy pattern that can impact how budgets are allocated and how services of the system are managed. They are interchangeable.

Participants:

Context - Government

Strategy - Policy

ConcreteStrategy - HealthPolicy, SafetyPolicy

Observer

Purpose:

Allows Citizens to respond to policy changes and availability of resources. They react to these changes by automatically updated changing their state of happiness. The city must have functioning utility systems that interact with buildings.

Why we used it:

This pattern supports a loosely coupled system where components can react to events without being tightly integrated. The Observer acts as a delivery system for the output of utilities (such as electricity, water, and waste and sewage management). The utility managers observe buildings for operational state changes and ensure the service delivery for buildings.

Participants:

Subject - Residential, Commercial, Industrial, Landmark, Service

ConcreteSubject - FactResidential, FactCommercial, FactIndstrial etc. (all types of concrete building classes)

Observer - UtilityManager

ConcreteObserver - UtilWaterSupply, UtilWasteMan, UtilSewageSyst

Ensures that citizens remain updated about utility changes, and this directly affects their state.

Composite

Purpose:

Traffic analysis and routing adjustments need to take place in our system, hence the Composite pattern is used for grouping individual roads into a single network of all roads that can be managed across the entire transportation network.

Why we used it:

The Composite pattern allows for hierarchical management of roads, making it easier to analyse and manage the transportation network from one location.

Participants:

Component - RoadComponent

Composite - RoadsComposite

Leaf - Highways, MainRoads, ResidentialStreets

Provides a structured way for the transportation system of our city to be managed.

Singleton

Purpose:

Many citizens, buildings and utilities need to access the CityMediator to communicate with their colleagues.

Why we used it:

By using a Singleton we have an easy access point from all classes that need to communicate with the CityMediator, as well as ensuring that there exists only one CityMediator at any time. Using this we were able to simplify constructors and mutators for several classes.

Participants:

Singleton - CityCentralMediator

Creates a single point of access for managing resources.

Iterator

Purpose:

It provides a way to access objects sequentially. Allows for the traversal of roads within the city, which is useful to monitor city connectivity.

Why we used it:

Makes it easier to navigate through road components and calculate routes. This would also allow us to navigate roads in different ways, such as regular iteration or pathfinding techniques such as Dijkstra's algorithm, which makes use of the iterator.

Participants:

Aggregate - RoadComponent

ConcreteAggregate - RoadsComposite

Iterator - RoadIterator

ConcreteIterator - RoadIteratorCon

Helps to manage roads effectively and to understand connectivity of the city.

Mediator

Purpose:

Citizens should respond dynamically to city conditions; thus, it tracks the creation of new buildings, citizens and utilities in the city.

Why we used it:

The City Mediator allows for centralised communication between objects, making it easier to manage interactions. It notifies citizens of important changes in utilities and services, and their satisfaction changes based on these changes.

Participants:

Mediator - CityMediator

ConcreteMediator - CityCentralMediator

Colleague - CityBlock

ConcreteColleague - UtilityManager, Building, RoadComponent

Ensures that changes in one component can be efficiently communicated to others in our system.

Chain of Responsibility

Purpose:

Requests related to taxes flow through the handlers until they can be addressed. Taxes need to be able to be set, adjusted, collected, and allocated to different sectors of the city.

Why we used it:

These functionalities are managed and processed by different handlers. By using this pattern, we are ensuring a structured approach to tax management.

Participants:

Handler - TaxHandler

ConcreteHandler - TaxRateHandler, TaxCollectionHandler, TaxAllocationHandler

Client - Government

Manages the processes related to tax collection and allocation

Visitor

Purpose:

Operations can be performed on objects without changing their classes. Different policies implemented by the government should influence the citizens of the city.

Why we used it:

The Visitor pattern facilitates these effects. It visits citizens and businesses and allows the government to monitor the impact of these policies on them.

Participants:

Client - Government

Visitor - TaxAndBudgetVisitor

ConcreteVisitor - CitizenTaxAB

Element - CityBlock

ConcreteElement - Citizen, Building

ObjectStructure - CityStructure

Manages the impact of policies on citizens.

Command

Purpose:

Encapsulates actions related to taxes into objects, allowing for parameterization of clients with queues, requests, and operations. This allows for better management and structured interactions when performing operations related to taxes.

Why we used it:

It structures our tax related requests through command objects, making it easier to carry out actions related to taxes in our city system.

Participants:

Command - TaxCommand

ConcreteCommand - SetTaxRateCommand, CollectTaxRateCommand, AllocateTaxRateCommand

Invoker - TaxManager

Organises tax operations within the simulation.

Facade

Purpose:

The Facade Pattern provides a simplified interface to a complex subsystem. It creates a unified point of access for multiple underlying components, which may otherwise have complex interactions or dependencies. The Facade Pattern is useful when you want to expose a single interface to manage different modules, reducing the learning curve for users of the system and making it easier to manage or extend functionality.

Why we used it:

In our city builder game, the Facade acts as the main game interaction point, providing access to essential gameplay actions like constructing buildings, gathering resources, and managing population growth. When a player initiates a new building project, for instance, the Facade might call multiple subsystems (e.g., resource check, building placement, and population assignment) without requiring the player or external code to interact with these subsystems individually.

Participants:

Facade - Game

Diagrams showing Implementation

State diagram:

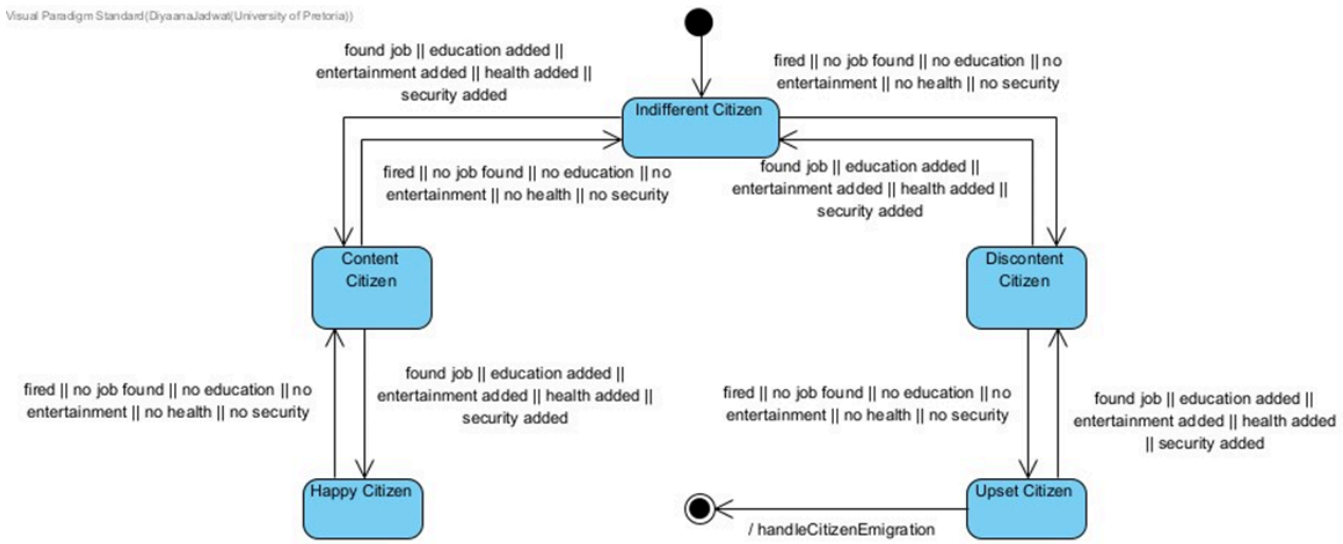


Figure 2: State diagram of Citizen Satisfaction Levels

Figure 2 helps to visualise how a citizen's mood changes based on events. If services are sufficient (basic healthcare, education, entertainment and security needs are met), or if the citizen acquires a job, they become increasingly happy. If they are unsatisfied with the services provided, or they lose their job or cannot find one, they become increasingly unhappy. When a citizen in the Upset state, they emigrate and are removed from the population count and no longer contribute to the city.

Activity diagram:

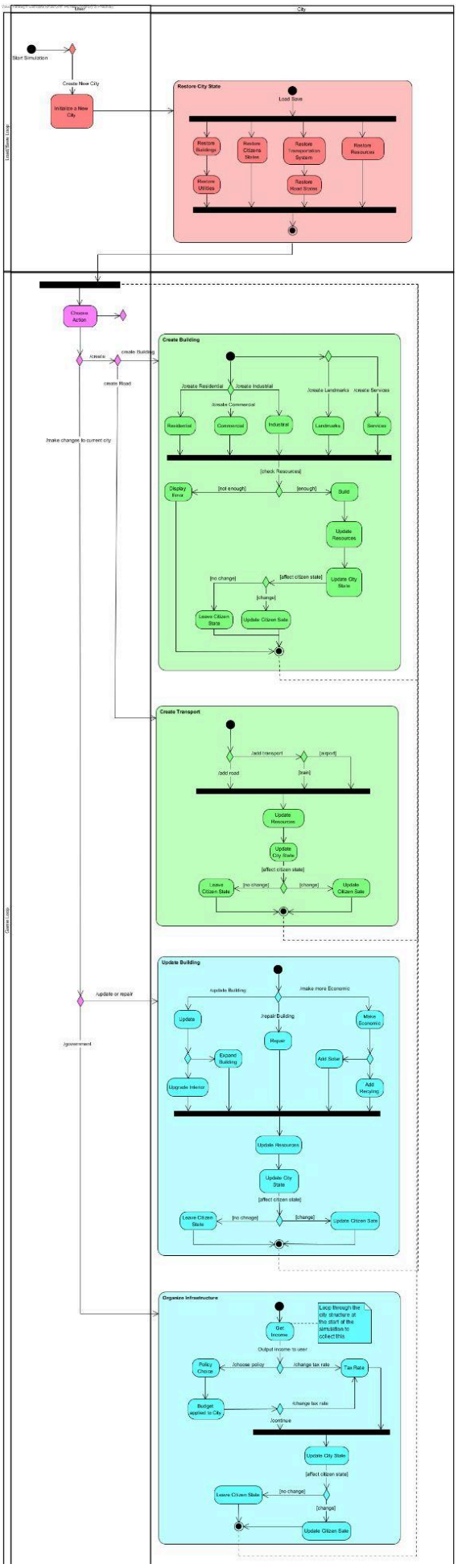


Figure 3: Activity Diagram of Simulation Engine

The game simulation starts by letting The user can either create a new city or load a saved city. After the city is loaded, the user may choose from many actions to update their city. These include: creating buildings, updating buildings, adding transport, organising infrastructure etc.

Creating a building allows the user to create a building from the different available types. It checks that resources are available to build the building before construction, then updates resources, the city state, and citizen states.

A user may also create transportation options or change existing routes. After doing so, resources, city states and citizen states are updated. A building may be updated and all affected elements of the city are updated.

Users may implement policies, set tax rates, allocate budgets etc. which all update the relevant parts of the system. When the user is ready to exit the simulation, the system saves the city state and the user progress for the next time they want to resume play.

Sequence diagrams:

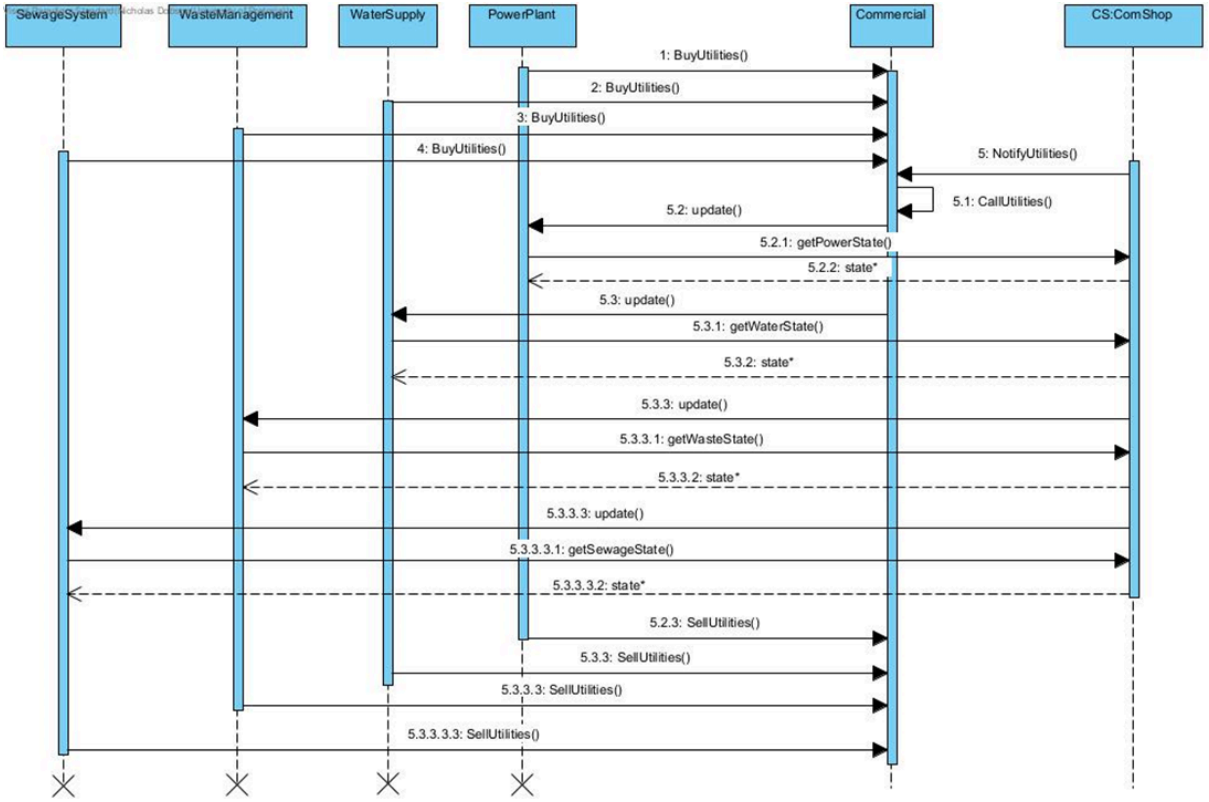


Figure 3: Sequence Diagram of Utility Management

This shows the interaction between the different utility systems and the manager in the system. Sewage System manages sewage state of the city. Waste Management handles waste. Water Supply supplies water to buildings, while Power Plant supplies electricity to them. Comercial is the entity that is consuming these resources. CS is a commercial shop that also consumes resources.

A commercial entity requests utilities through buyUtilities(), which sends a message to each utility provider. Each of them respond by confirming the availability of resources.

After they are bought, Commercial sends an update() request to change the state of their utilities. This verifies that the utilities and operating smoothly.
Once updated, all utilities are notified via notifyUtilities().

Once they are done, utilities are sold and resources are deallocated.

Utilities are monitored to check that they are available and ready for use.

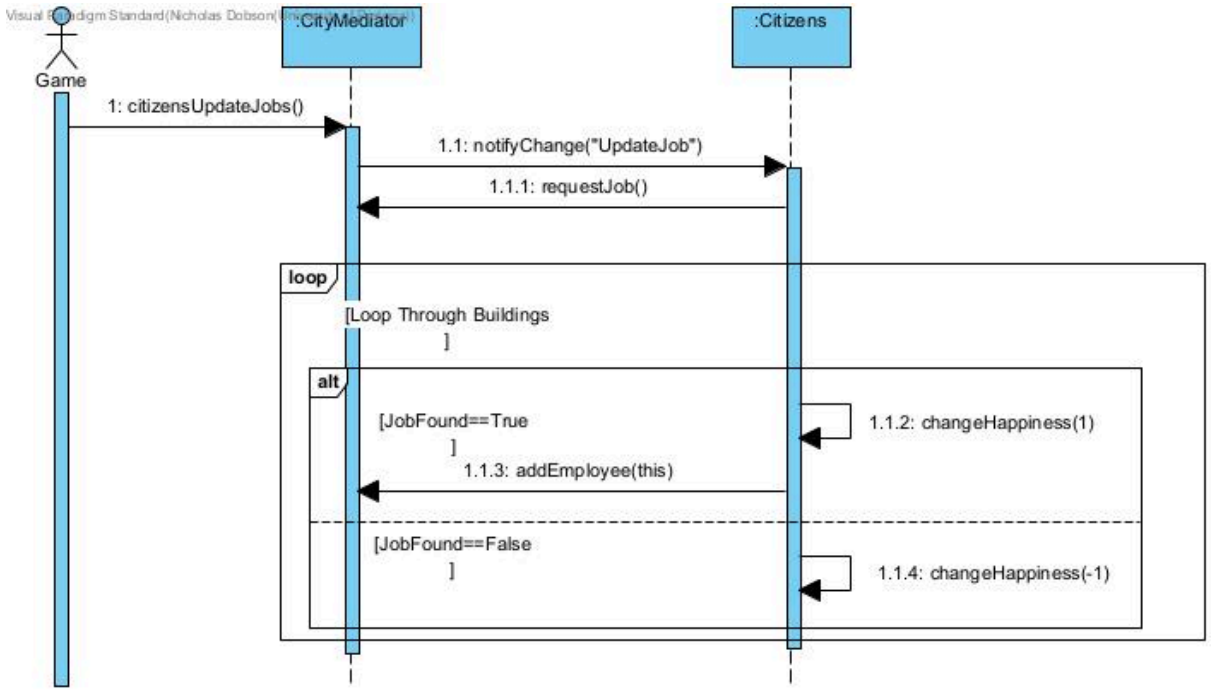


Figure 4: Sequence diagram of Citizen Mediator

Figure 4 and 5 display similar processes regarding the Mediator. Game first updates all jobs for citizens on the CityMediator, which notifies all citizens to update their job status and handles the requests for available jobs. The mediator checks if any buildings have available jobs, if it is found, the citizen is employed and their happiness increases. If no job is found (Figure 4), their happiness decreases.

Communication diagram:

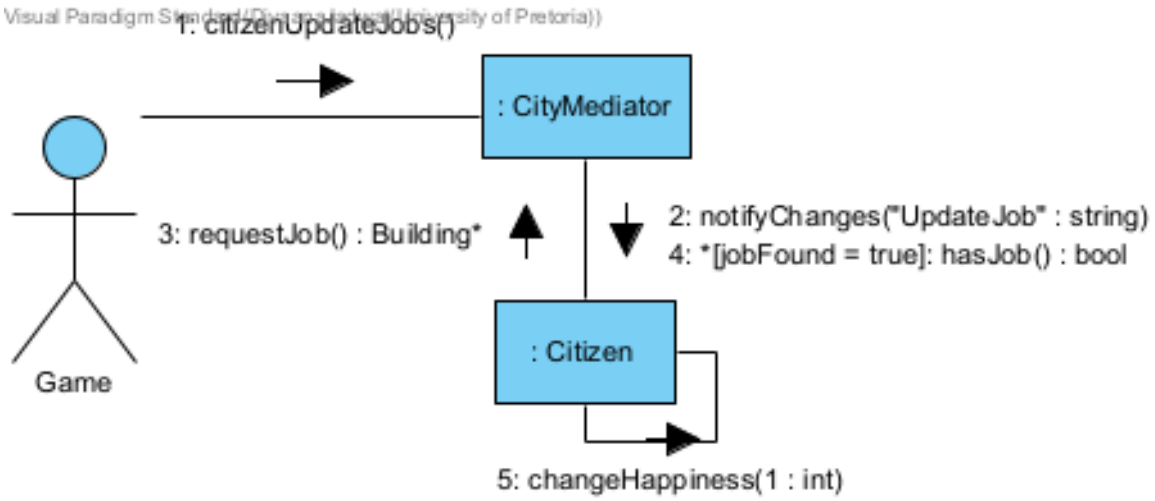


Figure 5: Communication diagram of Citizen Mediator when a job is found

Figure 5 shows the same process as Figure 4, but rather displays the situation where a citizen is successful in their search for a job, and their happiness increases.

Object diagram:

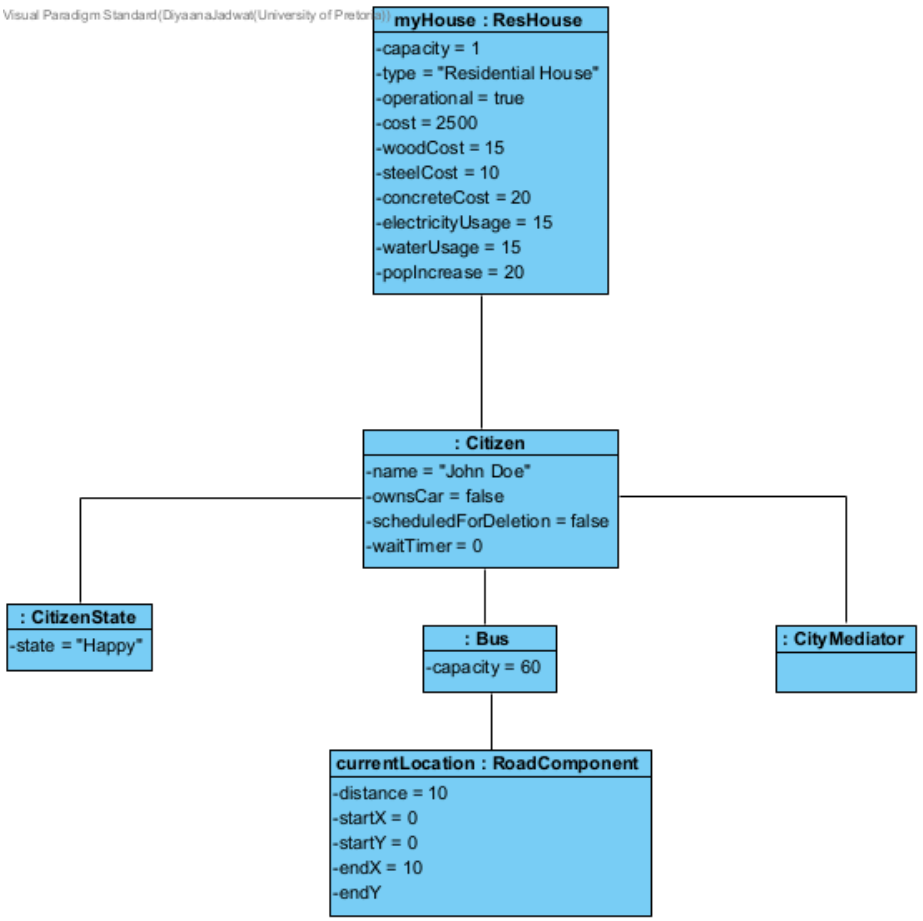


Figure 6: Object Diagram of a happy citizen travelling on bus

The Citizen represents an individual in the simulation. They reside at a Residential House. The CitizenState object holds the citizen's current emotional state, which is currently Happy. They are currently on the bus. The currentLocation attribute keeps track of the bus's current position and the bus calculates its route based on the Citizen's desired destination. City Mediator facilitates communication between citizens, transport, buildings and other entities within the system.

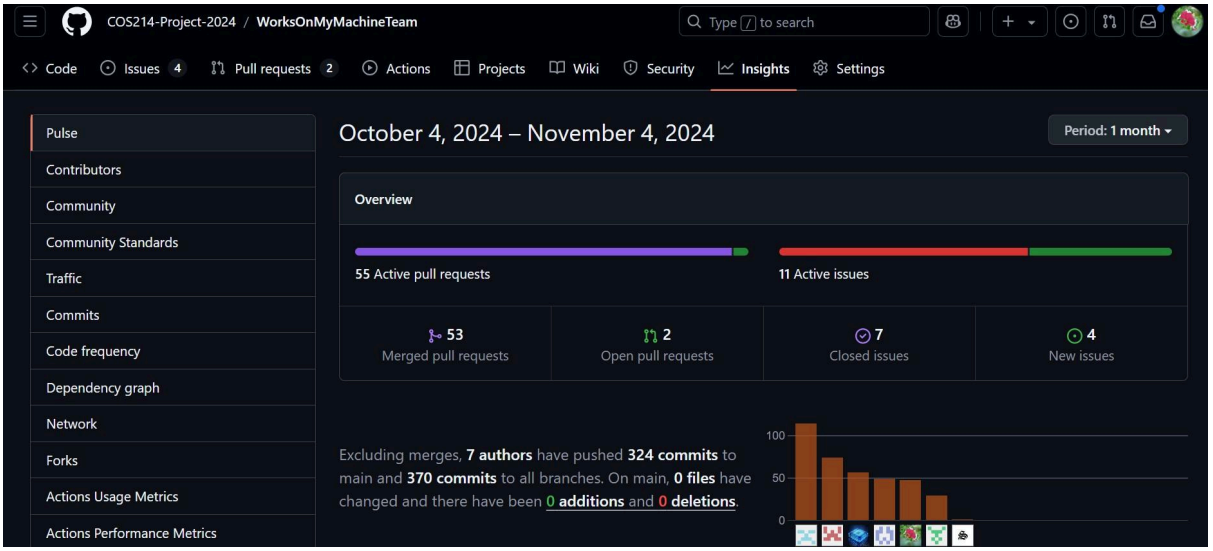
Version Control with Git:

The main branch is where our production code is. It is the most stable branch and our final product. Only finished code that has been tested fully is added to main.

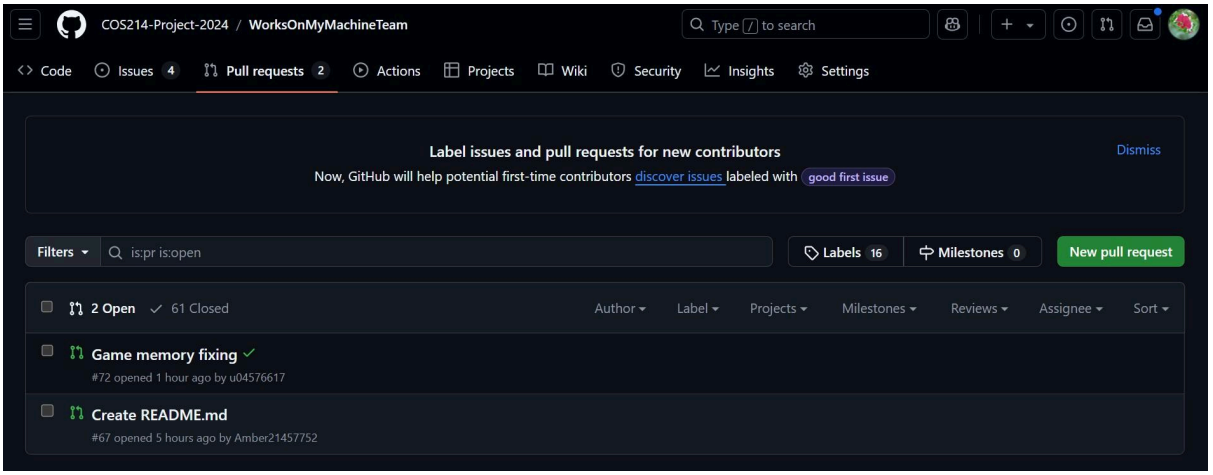
The dev branch is the intermediary between main and our other branches. All features are merged to dev first and tested before merging to main. It is a fairly stable branch.

We then created branches per each section of our system as we worked. They are specific to certain parts of the system and each perform unique tasks. These branches are merged to dev once ready.

Commits

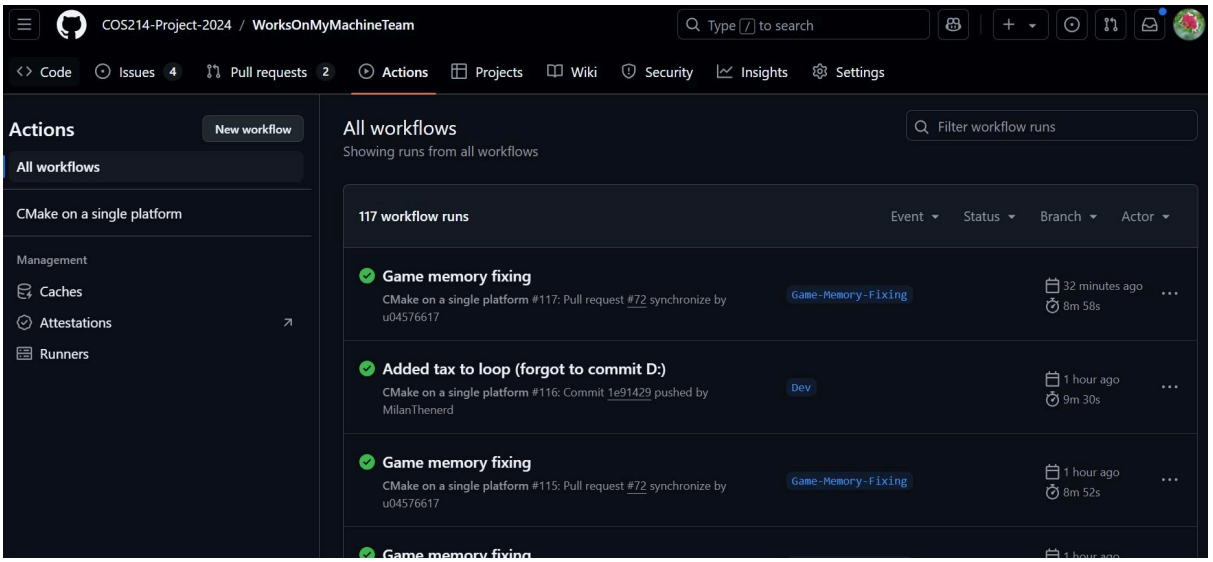


Pull requests



GitHub Actions

We made use of GitHub Actions to perform automated unit testing.



References

1. Huxley, M. (2015). *Urban Planning - an overview | ScienceDirect Topics*. [online] Sciencedirect.com. Available at: https://www.sciencedirect.com/topics/social-sciences/urban-planning.

2. Lai, S-K. (2017). *City Management: Theories, Methods, and Applications (Book Proposal).* [online] researchgate.net. Available at: https://www.researchgate.net/publication/322152661_City_Management_Theories_Methods_and_Applications_Book_Proposal

3. Principles for Better Cities I Principles for Better Cities Towards Sustainable Development in Metropolitan Regions, Precincts and Places. (n.d.). Available at: https://www.metropolis.org/sites/default/files/resources/Principles-for-Better-Cities.pdf.

4. Saliba, S. (2018). *Ten urban planning principles every humanitarian should know*. [online] International Institute for Environment and Development. Available at: https://www.iied.org/ten-urban-planning-principles-every-humanitarian-should-know.

5. Archi_com (2022). *THE ELEMENTS OF A CITY ⋆ Archi-Monarch*. [online] Archi-Monarch. Available at: https://archi-monarch.com/the-elements-of-a-city/.