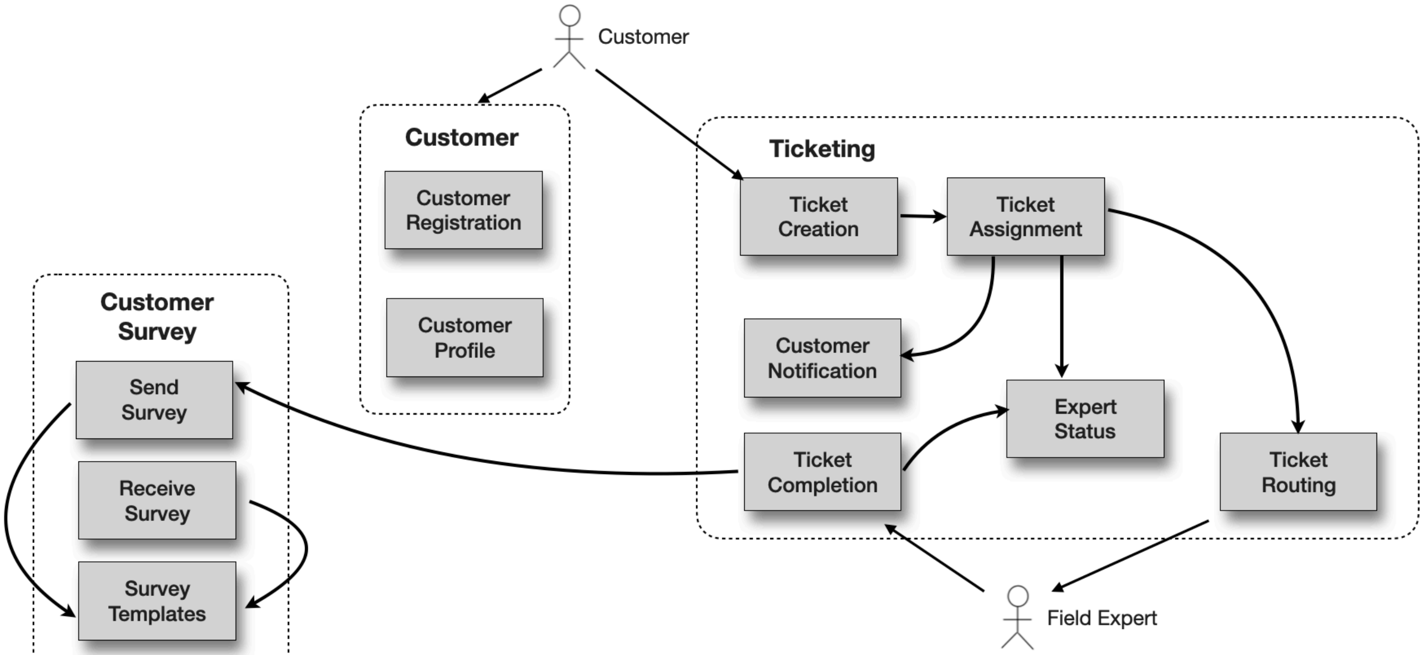
Architecture Definition Language

Architecture Definition Language (ADL)

Pseudo-code for describing and governing the structure of a system



Architecture Definition Language (ADL)

Pseudo-code for describing and governing the structure of a system (its logical architecture)

Meta Data

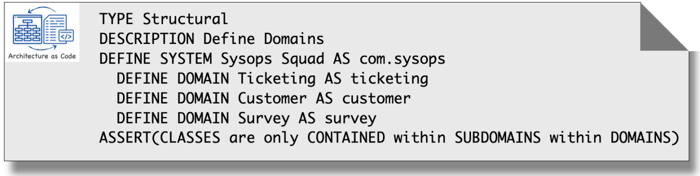
```
REQUIRES text
DESCRIPTION text
PROMPT text
```

Artifacts

```
SYSTEM logical_name AS physical_name
DOMAIN logical_name AS physical_name
SUBDOMAIN logical_name AS physical_name
COMPONENT logical_name AS physical_name
LIBRARY logical_name AS library_name
SERVICE logical_name AS physical_name
CONST variable_name AS value
```

Actions

```
ASSERT (some condition is met)
FOREACH $X IN list of X DO actions END
    CLASSES
    DOMAINS
    SUBDOMAINS
    COMPONENTS
    SERVICES
    CONTAINED WITHIN
    CONTAINS
    DEPEND ON / DEPENDS ON / DEPENDENCY ON
```


Architecture as Code

```
TYPE Structural
DESCRIPTION Define Domains
DEFINE SYSTEM Sysops Squad AS com.sysops
  DEFINE DOMAIN Ticketing AS ticketing
  DEFINE DOMAIN Customer AS customer
  DEFINE DOMAIN Survey AS survey
ASSERT(CLASSES are only CONTAINED within SUBDOMAINS within DOMAINS)
```

ArchUnit

TSArch

It's a library for checking architecture conventions in TypeScript&JavaScrip
You check dependencies between files, folders and slices, check for cyclic
ArchUnit but for TS/JS projects.

NetArchTest

Azure Pipelines succeeded

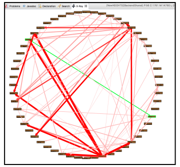
A fluent API for .Net Standard that can enforce architectural rules

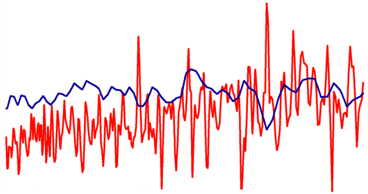
ArchUnit

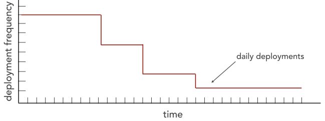
ArchUnitNET build passing License Apache 2.0

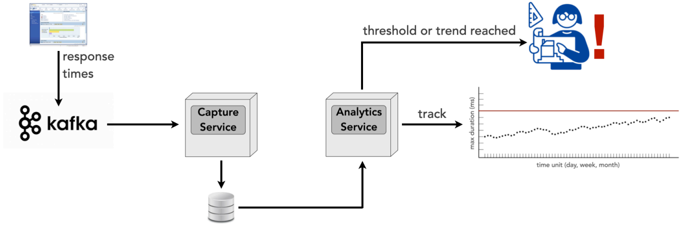
Welcome to PyTestArch

pytestarch 2.0.3









Architecture Definition Language (ADL)

Pseudo-code for describing and governing the structure of a system

Meta Data

```
REQUIRES text
DESCRIPTION text
PROMPT text
```

Artifacts

```
SYSTEM logical_name AS physical_name
DOMAIN logical_name AS physical_name
SUBDOMAIN logical_name AS physical_name
COMPONENT logical_name AS physical_name
LIBRARY logical_name AS library_name
SERVICE logical_name AS physical_name
CONST variable_name AS value
```

Actions

```
ASSERT (some condition is met)
FOREACH $X IN list of X DO actions END
    CLASSES
    DOMAINS
    SUBDOMAINS
    COMPONENTS
    SERVICES
    CONTAINED WITHIN
    CONTAINS
    DEPEND ON / DEPENDS ON / DEPENDENCY ON
```

Architecture Definition Language (ADL)

REQUIRES `text`

Provides a text-based meta-data description of what artifacts are needed for this test to run (log files, ArchUnit, NetArchTest, lint tools, custom observability metrics, streaming broker, etc.), as well as any code decoration (annotations, custom attributes, interfaces, etc.). Note: When generating code using an LLM, remove this meta data from the prompt.

Usage:

```
REQUIRES Generated ArchUnit code via ChapGPT
```

Architecture Definition Language (ADL)

DESCRIPTION `text`

Provides a text-based meta-data description of the this test. Note: When generating code using an LLM, remove this meta data from the prompt.

Usage:

```
   DESCRIPTION Define Ticketing Components
```

Architecture Definition Language (ADL)

```
PROMPT text
```

Provides a text-based meta-data description of the LLM prompt used to generate source code based on this pseudo-code test.

Usage:

```
PROMPT Based on this pseudo-code, write a ArchUnit test in Java
```

Architecture Definition Language (ADL)

```
SYSTEM logical_name AS physical_name
```

Defines the system context for the architectural test, where the *physical_name* represents a root directory or namespace for the system and the *logical_name* represents a human-readable name of the system.

Usage:

```
DEFINE SYSTEM Sysops Squad AS com.sysops
```

Architecture Definition Language (ADL)

```
DOMAIN logical_name AS physical_name
```

Defines the domain context for the architectural test, where the *physical_name* represents a high-level directory or namespace for the domain and the *logical_name* represents a human-readable name of the domain.

Usage:

```
DEFINE DOMAIN Ticketing AS ticketing
```

Architecture Definition Language (ADL)

```
SUBDOMAIN logical_name AS physical_name
```

Defines the subdomain context for the architectural test, where the *physical_name* represents a mid-level directory or namespace for the subdomain and the *logical_name* represents a human-readable name of the subdomain.

Usage:

```
DEFINE SUBDOMAIN Ticket Assignment AS assignment
```

Architecture Definition Language (ADL)

```
COMPONENT logical_name AS physical_name
```

Defines a component for the architectural test, where the *physical_name* represents a leaf-level directory or namespace for the component and the *logical_name* represents a human-readable name of the component.

Usage:

```
    DEFINE Component Ticket Routing AS routing
```

Architecture Definition Language (ADL)

```
LIBRARY logical_name AS library_name
```

Defines a shared or third-party library for the architectural test, where the *physical_name* represents the name of the library artifact and the *logical_name* represents a human-readable name of the library.

Usage:

```
DEFINE LIBRARY Security Library AS ReallyGoodSecurity.DLL
```

Architecture Definition Language (ADL)

```
SERVICE logical_name AS physical_name
```

Defines a service (separately deployed unit of software) for the architectural test, where the *physical_name* represents a high-level directory or namespace describing the service boundary and the *logical_name* represents a human-readable name of the service.

Usage:

```
   DEFINE SERVICE Ticket Creation Service AS creation_service
```

Architecture Definition Language (ADL)

```
CONST variable_name AS value
```

Defines a constant value used within an ASSERT of FOREACH action.

Usage:

```
DEFINE CONST TIME-INTERVAL AS 2 DAYS
```

Architecture Definition Language (ADL)

```
ASSERT (some condition is met)
```

Enforces a particular constraint on the defined artifacts using keywords (verbs), written in text format.

Usage:

```
ASSERT(survey has NO DEPENDENCY on {ticketing, reporting})
```

Architecture Definition Language (ADL)

```
FOREACH $X IN list of X DO actions END
```

Enforces a particular constraint on the defined artifacts using keywords (verbs), written in text format.

Usage:

```
FOREACH $X in COMPONENTS DO
   ASSERT($X has NO DEPENDENCY on ticketing)
```

Architecture Definition Language (ADL)

CLASSES

Used within the context of an ASSERT or FOREACH action to describe a collection of classes within a system, domain, subdomain, component, library, or service.

Usage:

```
   ASSERT(CLASSES are only CONTAINED within DOMAINS)
```

Architecture Definition Language (ADL)

DOMAINS

Used within the context of an ASSERT or FOREACH action to describe a collection of defined domains within a system, library, or service.

Usage:

```
ASSERT(CLASSES are only CONTAINED within DOMAINS)
```

Architecture Definition Language (ADL)

SUBDOMAINS

Used within the context of an ASSERT or FOREACH action to describe a collection of defined subdomains within a domain, library, or service.

Usage:

```
ASSERT(CLASSES are only CONTAINED within SUBDOMAINS)
```

Architecture Definition Language (ADL)

COMPONENTS

Used within the context of an ASSERT or FOREACH action to describe a collection of defined components within a domain, subdomain, library, or service.

Usage:

```
ASSERT(admin has NO DEPENDENCY on other COMPONENTS)
```

Architecture Definition Language (ADL)

SERVICES

Used within the context of an ASSERT or FOREACH action to describe a collection of defined services within a system, domain, or subdomain.

Usage:

```
ASSERT(admin_service has NO DEPENDENCY on other SERVICES)
```

Architecture Definition Language (ADL)

CONTAINED WITHIN

Verb used within the context of an ASSERT or FOREACH action to describe a defined artifact that is contained within another higher-level artifact.

Usage:

```
ASSERT(CLASSES are exclusively CONTAINED WITHIN COMPONENTS)
```

Architecture Definition Language (ADL)

CONTAINS

Verb used within the context of an ASSERT or FOREACH action to describe a defined artifact that contains other lower-level artifacts.

Usage:

```
ASSERT(ticketing_service CONTAINS security_library)
```

Architecture Definition Language (ADL)

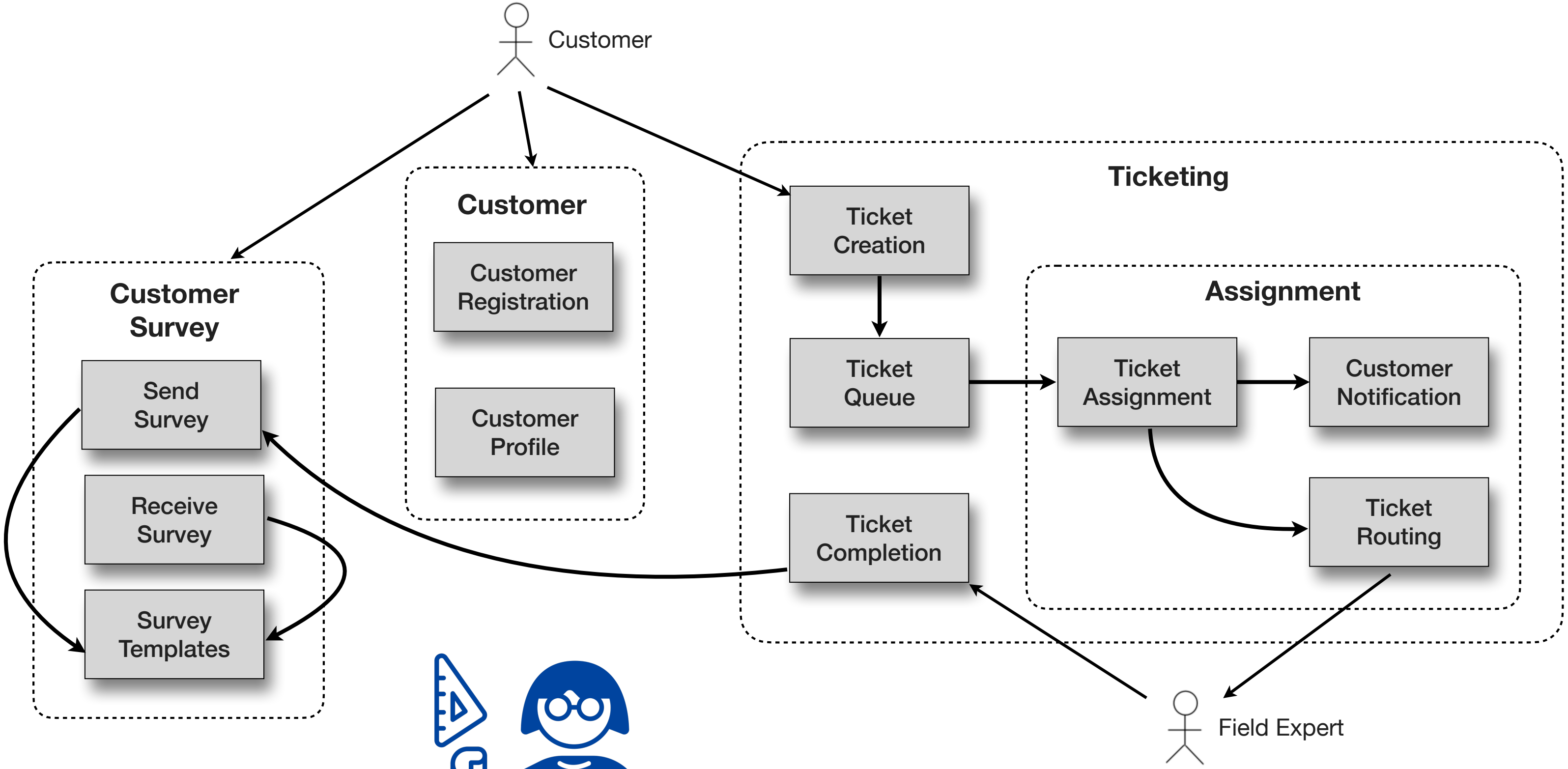
DEPEND ON / DEPENDS ON / DEPENDECY ON

Verb used within the context of an ASSERT or FOREACH action to describe a defined artifact that depends other same-level or higher-level artifacts.

Usage:

```
  ASSERT(survey has NO DEPENDENCY on {ticketing, reporting})
```

Example



Customer

Customer Survey

Send Survey

Receive Survey

Survey Templates

Customer

Customer Registration

Customer Profile

Ticketing

Ticket Creation

Ticket Queue

Ticket Completion

Assignment

Ticket Assignment

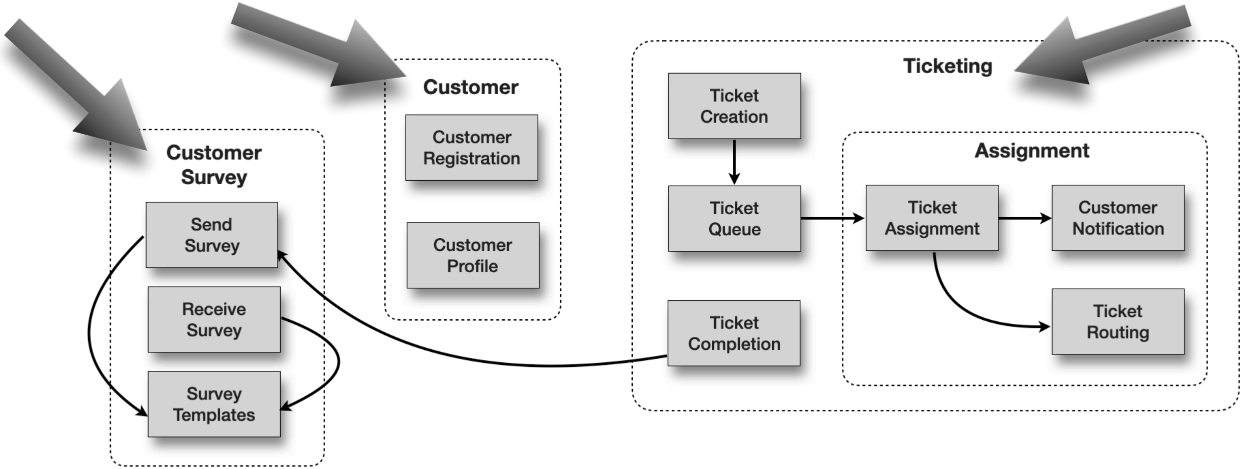
Customer Notification

Ticket Routing

Field Expert

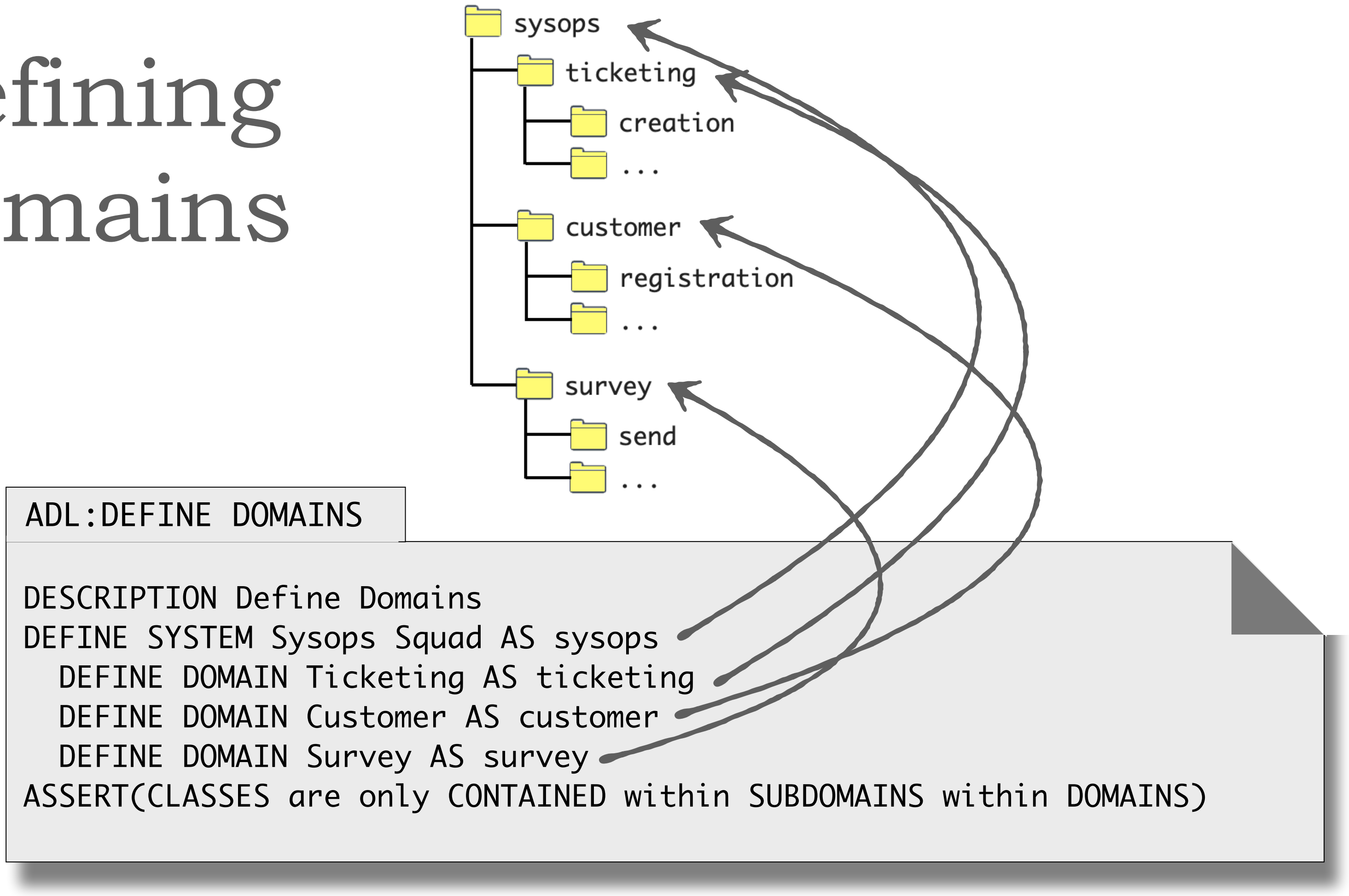
software architect

Defining Domains



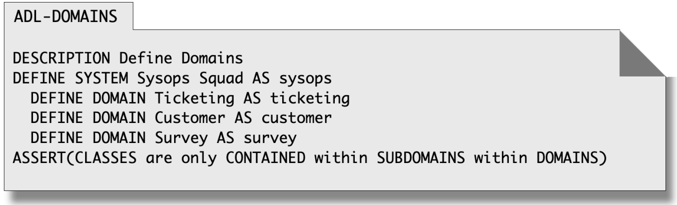
```
ADL:DEFINE DOMAINS

DESCRIPTION Define Domains
DEFINE SYSTEM Sysops Squad AS sysops
  DEFINE DOMAIN Ticketing AS ticketing
  DEFINE DOMAIN Customer AS customer
  DEFINE DOMAIN Survey AS survey
ASSERT(CLASSES are only CONTAINED within SUBDOMAINS within DOMAINS)
```

Defining Domains



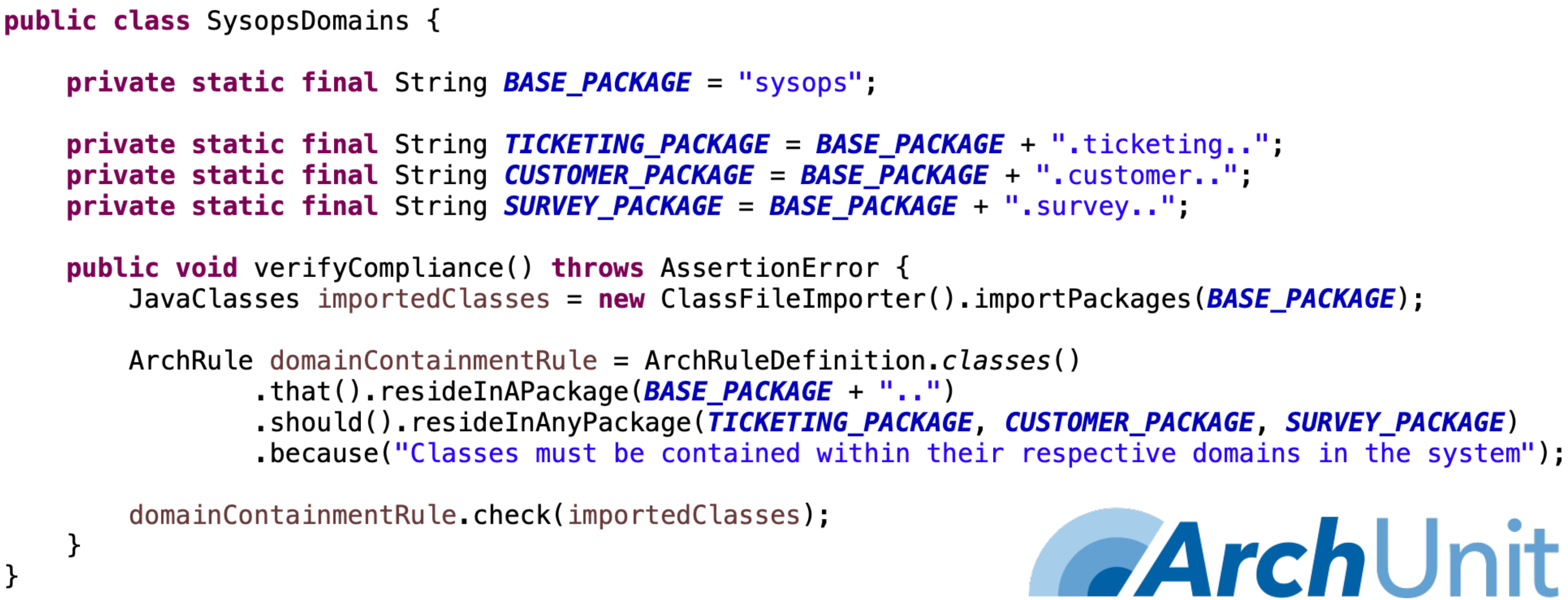
ADL:DEFINE DOMAINS

```
DESCRIPTION Define Domains
DEFINE SYSTEM Sysops Squad AS sysops
  DEFINE DOMAIN Ticketing AS ticketing
  DEFINE DOMAIN Customer AS customer
  DEFINE DOMAIN Survey AS survey
ASSERT(CLASSES are only CONTAINED within SUBDOMAINS within DOMAINS)
```



```
ADL-DOMAINS

DESCRIPTION Define Domains
DEFINE SYSTEM Sysops Squad AS sysops
  DEFINE DOMAIN Ticketing AS ticketing
  DEFINE DOMAIN Customer AS customer
  DEFINE DOMAIN Survey AS survey
ASSERT(CLASSES are only CONTAINED within SUBDOMAINS within DOMAINS)
```

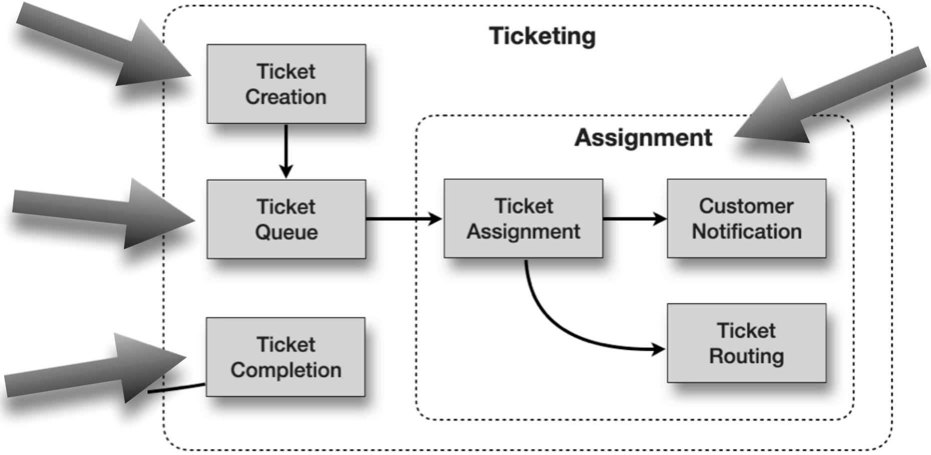
OpenAI
GPT-4o



```java
public class SysopsDomains {

    private static final String BASE_PACKAGE = "sysops";

    private static final String TICKETING_PACKAGE = BASE_PACKAGE + ".ticketing..";
    private static final String CUSTOMER_PACKAGE = BASE_PACKAGE + ".customer..";
    private static final String SURVEY_PACKAGE = BASE_PACKAGE + ".survey..";

    public void verifyCompliance() throws AssertionError {
        JavaClasses importedClasses = new ClassFileImporter().importPackages(BASE_PACKAGE);

        ArchRule domainContainmentRule = ArchRuleDefinition.classes()
                .that().resideInAPackage(BASE_PACKAGE + "..")
                .should().resideInAnyPackage(TICKETING_PACKAGE, CUSTOMER_PACKAGE, SURVEY_PACKAGE)
                .because("Classes must be contained within their respective domains in the system");

        domainContainmentRule.check(importedClasses);
    }
}
```

ArchUnit

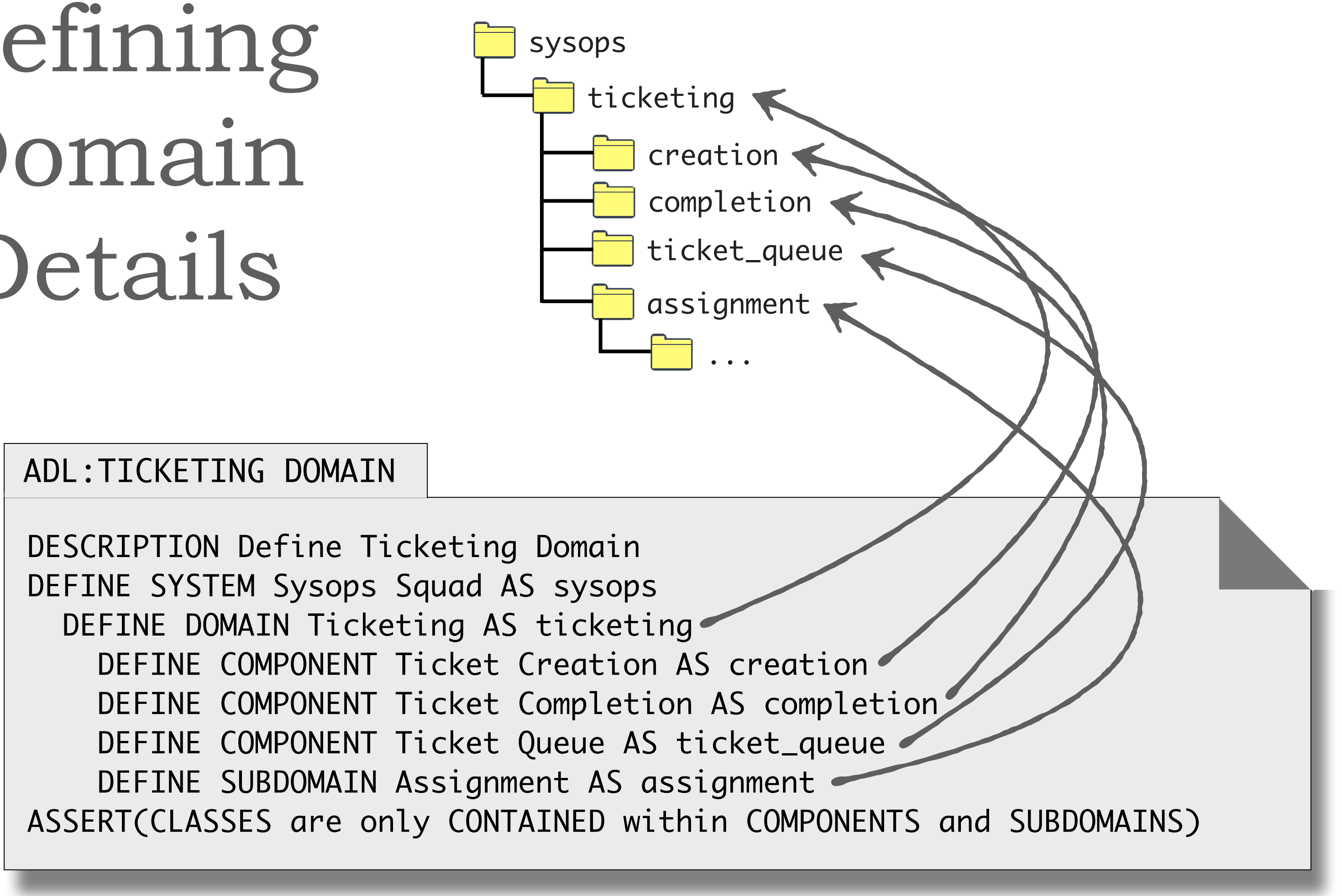
Defining Domain Details



ADL:TICKETING DOMAIN

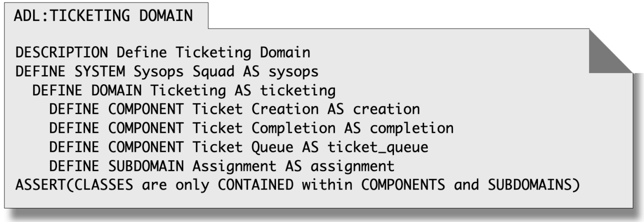
```
DESCRIPTION Define Ticketing Domain
DEFINE SYSTEM Sysops Squad AS sysops
  DEFINE DOMAIN Ticketing AS ticketing
    DEFINE COMPONENT Ticket Creation AS creation
    DEFINE COMPONENT Ticket Completion AS completion
    DEFINE COMPONENT Ticket Queue AS ticket_queue
    DEFINE SUBDOMAIN Assignment AS assignment
ASSERT(CLASSES are only CONTAINED within COMPONENTS and SUBDOMAINS)
```

Defining Domain Details



ADL:TICKETING DOMAIN

```
DESCRIPTION Define Ticketing Domain
DEFINE SYSTEM Sysops Squad AS sysops
  DEFINE DOMAIN Ticketing AS ticketing
    DEFINE COMPONENT Ticket Creation AS creation
    DEFINE COMPONENT Ticket Completion AS completion
    DEFINE COMPONENT Ticket Queue AS ticket_queue
    DEFINE SUBDOMAIN Assignment AS assignment
ASSERT(CLASSES are only CONTAINED within COMPONENTS and SUBDOMAINS)
```



```
ADL:TICKETING DOMAIN

DESCRIPTION Define Ticketing Domain
DEFINE SYSTEM Sysops Squad AS sysops
  DEFINE DOMAIN Ticketing AS ticketing
    DEFINE COMPONENT Ticket Creation AS creation
    DEFINE COMPONENT Ticket Completion AS completion
    DEFINE COMPONENT Ticket Queue AS ticket_queue
    DEFINE SUBDOMAIN Assignment AS assignment
ASSERT(CLASSES are only CONTAINED within COMPONENTS and SUBDOMAINS)
```

OpenAI GPT-4o

```java
public class TicketingDomain {

    private static final String BASE_PACKAGE = "sysops.ticketing";

    private final JavaClasses importedClasses =
            new ClassFileImporter().importPackages(BASE_PACKAGE);

    public void verifyCompliance() {

         ArchRule rule = ArchRuleDefinition.classes()
            .that().resideInAPackage(BASE_PACKAGE + "..")
            .should().resideInAnyPackage(
                BASE_PACKAGE + ".creation..",
                BASE_PACKAGE + ".ticket_queue..",
                BASE_PACKAGE + ".completion..",
                BASE_PACKAGE + ".assignment.."
            )
            .because("All classes in the 'ticketing' domain must be inside " +
                     "defined components or subdomains");

        rule.check(importedClasses);
    }
}
```

ArchUnit