Example from Pill Dronz

- The test basis:
  - The *extend* feature shall generate a new path or will fail to generate a new path
  - The feature shall accept a path *r* and a point *p* and return a path *r'* whose end point is *p* and the point preceding the end point of *r'* is the end point of *r*, provided:
    - The distance from the end point of *r* to *p* is not too long.
    - The step from the end point of *r* to *p* does not pass through a no-fly zone
  - If the preceding conditions are not satisfied, then the feature generates an error.
- Test completion criterion:
  - 100% equivalence partition coverage is achieved with all tests passed.

Identify Independently Testable Features

- The *extend* feature
- Parameter: *r* a path represented as a list of points
- Parameter: *p* a point
- Environment Element: *No Fly* a list of pairs of points representing rectangles

Identify relevant value classes

- Path *r*
  - Null
  - Length 1
  - Length > 1 with adjacent points acceptably spaced
  - Length > 1 with at least one successive pairs of points too far apart
  - Length is max if there is a limit or perhaps just very long
- Point *p*
  - Well Formed
  - Lat component is erroneous
  - Long component is erroneous

Identify relevant value classes

- Environment Element *No Fly:*
  - Null
  - Length 1
  - Length > 1 and all elements are well-formed
  - Length > 1 and at least one element is ill-formed
  - Length > 1 and at least one element represents a no-fly zone containing the final point of *r*
  - Length > 1 and at least one element represents a no-fly zone containing the point *p*
  - Length > 1 and at least one element represents a no-fly zone that intersects with the line from the final point of *r* to *p*
  - *We could extend this further*

Test Case Specifications

- The combinations of value classes from each for each parameter and environment element makes a test case **specification**.
- For example: *p* – a valid point, *r* – a one element list, *NoFly* – a very long list
- Or: p - malformed lat component, *r* – a well-formed path, *NoFly* – null
- We have 5 x 3 x 7 = 105 test case specifications for this very simple example.
- In more complex systems the number of test cases becomes unmanageable

Error Constraints

- These are added to value classes and help reduce the number of test case specifications.
- Path *r*
  - Null
  - Length 1
  - Length > 1 with adjacent points acceptably spaced
  - Length > 1 with at least one successive pairs of points too far apart [error]
  - Length is max if there is a limit or perhaps just very long
- Point *p*
  - Well Formed
  - Lat component is erroneous [error]
  - Long component is erroneous [error]

Error Constraints

- If a value class contains only erroneous values we label it with an [error] constraint.
- We assume that test case specifications containing a value class with an [error] label will not test the main part of the code so we reduce the number of test case specifications to one.
- For example: there are 35 test case specifications that include a Lat malformed value class and a Long malformed value class so we can replace 70 test case specifications with 2.
- We also have the [single] constraint that indicates a value class we only want to include in one test case specification (mainly to cut down on number of test case specifications)

Property Constraints

- Constraint [property] [if-property] rule out invalid combinations of values. This is looking at constraining combinations across categories.
- [property] groups values of a single parameter to identify subsets of values with common properties
- [if-property] bounds the choices of values for a category that can be combined with a particular value selected for a different category

Example Property Constraint

- Environment Element *No Fly:*
  - …
  - Length > 1 and all elements are well-formed [GOOD]
  - …
- Point *p*
  - Well Formed [if GOOD]
  - Lat component is erroneous
  - Long component is erroneous
- This would constrain the test case specifications only to consider testing a well-formed point with well-formed *No Fly* lists.