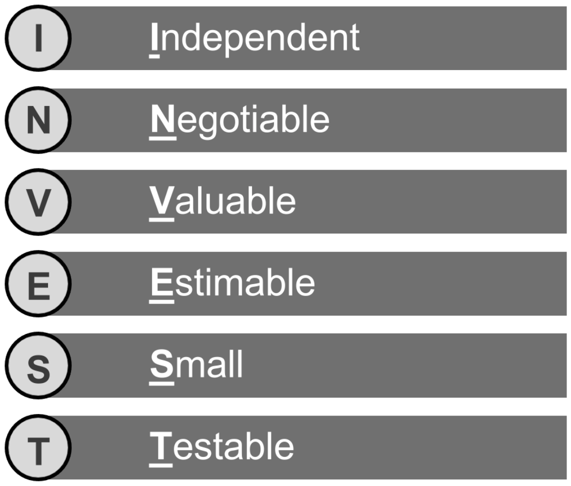
Appendix A

Writing and Splitting Stories

A.1 INVEST in a Good Story

INVEST in a good Story



Stories are Independent

- Write closed Stories
- Slice through the architecture (vertical)
- Write only the delta (the change)
- Remove non-value dependencies (both technical and functional)

Non-value dependencies

As an administrator, I can set the consumer's password security rules so that users are required to create and retain secure passwords, keeping the system secure.

As a consumer, I am required to follow the password security rules set by the administrator so that I can maintain high security for my account.

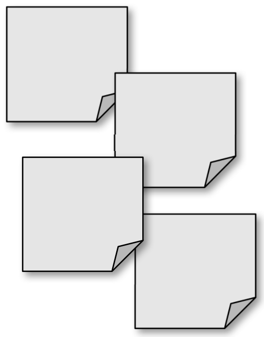
Split in different manner: *setup* and *enforcement* in each Story

As an administrator, I can set the password expiration period so that users are forced to change their passwords periodically.

As an administrator, I can set the password strength characteristics so that users are required to create passwords that are difficult to hack.

Stories are **N**egotiable

- User stories are statements of *intent*, not contracts or detailed requirements
- Too much detail gives impression of false precision or completeness
- Flexibility drives release schedule and goals



Stories are **V**alued by users

- Write Stories in the voice of the Customer
- Write for one user

“Technical” voice and value

Refactor the error logging system

User voice and value

As a consumer, I can receive a consistent and clear error message anywhere in the product so that I know how to address the issue.

As a technical support member, I want the user to receive a consistent and clear message anywhere in the application so they can fix the issue without calling support.

Stories are **E**stimable

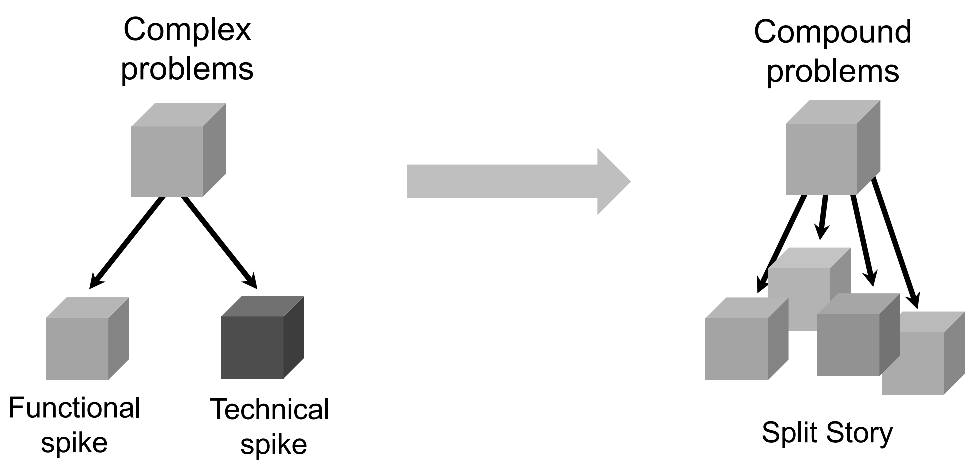
User stories are for planning and tracking

- To measure release progress, each Story needs an estimate of size

Estimating may be difficult because …

- Developers lack the domain knowledge to know what is to be done
- Developers lack the technical knowledge to know how to do something
- The Story is too big or too vague

Stories are **S**mall enough to fit in iterations



Stories are **T**estable

- Write Stories that are testable
- Include acceptance criteria for each Story

Not testable	*Testable*
As a power generation company salesperson, I want my search results to return quickly so that I can find relevant contacts for the information I am searching.	As a power generation company salesperson, I want to receive the first page of search results within 3 seconds so that I can find relevant contacts quickly.

A.2 Splitting Features and Stories

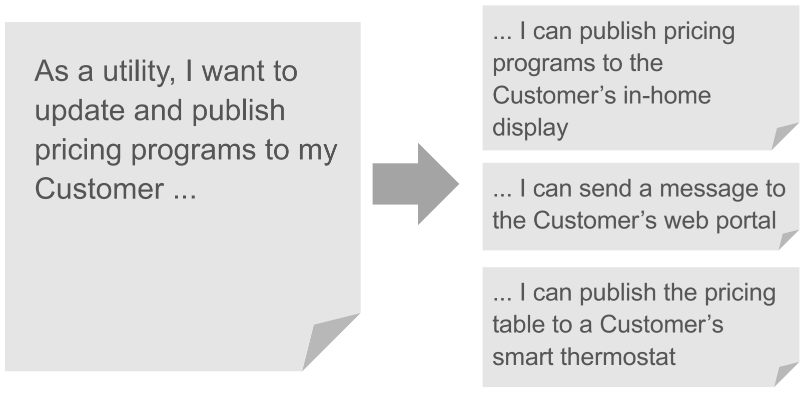
Splitting Features and Stories

Techniques for splitting Features and Stories to fit within their boundaries (PI and Iteration respectively)

1. Work flow steps	6. Data methods
2. Business rule variations	7. Defer system qualities
3. Major effort	8. Operations
4. Simple/complex	9. Use-case scenarios
5. Variations in data	10. Break out a spike

1. Split by work flow steps

Identify specific steps that a user takes to accomplish a work flow, then implement the work flow in increments.



As a utility, I want to update and publish pricing programs to my Customer ...

... I can publish pricing programs to the Customer's in-home display

... I can send a message to the Customer's web portal

... I can publish the pricing table to a Customer's smart thermostat

2. Split by business rule variations

Business rule variations often provide a straightforward splitting scheme.

As a utility, I can sort Customers by different demographics ...

- ... sort by zip code
- ... sort by home demographics
- ... sort by energy consumption

3. Split by major effort

Split into several parts, with the first requiring the most effort. More functionality can be added later on.

As a user, I want to be able to select/change my pricing program with my utility through my web portal ...

- ... I want to use time-of-use pricing
- ... I want to prepay for my energy
- … I want to enroll in critical peak pricing

4. Split by simple/complex

Simplify! What’s the simplest version that can possibly work?

As a user, I basically want a fixed price, but I also want to be notified of critical peak pricing events ...

→

... respond to the time and the duration of the critical peak pricing event

... respond to emergency events

5. Split by variations in data

Variations in data provide additional opportunities, such as those shown in this localization example.

As a utility, I can send messages to Customers ...

→

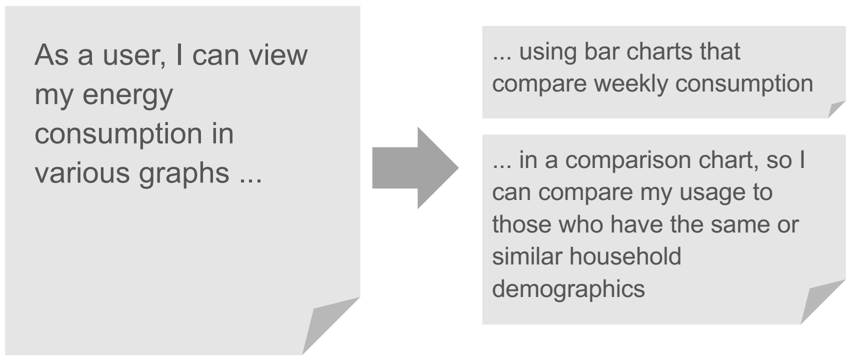
Customers who want their messages in Spanish

Customers who want their messages in Arabic

Customers who want their messages in … etc.

6. Split by data methods

Complexity can be in the interface rather than the functionality itself. Split these Stories to build the simplest interface first.



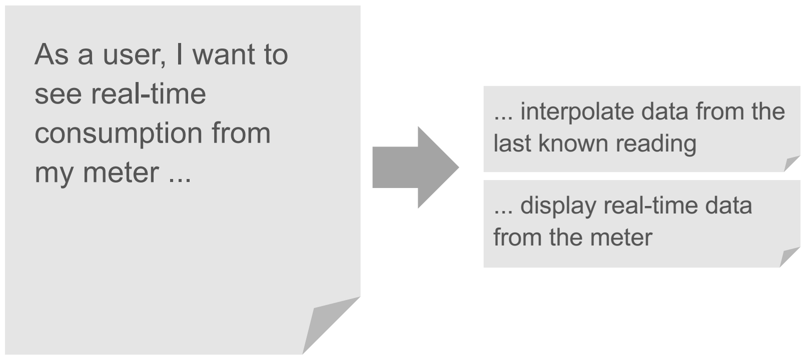
As a user, I can view my energy consumption in various graphs ...

... using bar charts that compare weekly consumption

... in a comparison chart, so I can compare my usage to those who have the same or similar household demographics

7. Split by deferring system qualities

Sometimes functionality isn’t that difficult. More effort may be required to make it faster ... or more precise ... or more scalable.



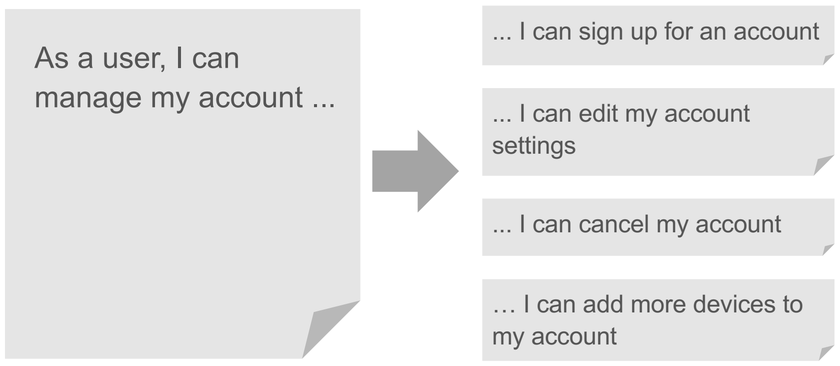
As a user, I want to see real-time consumption from my meter ...

... interpolate data from the last known reading

... display real-time data from the meter

8. Split by operations

Split by type of operation: Create Read Update Delete (CRUD)



As a user, I can manage my account ...

... I can sign up for an account

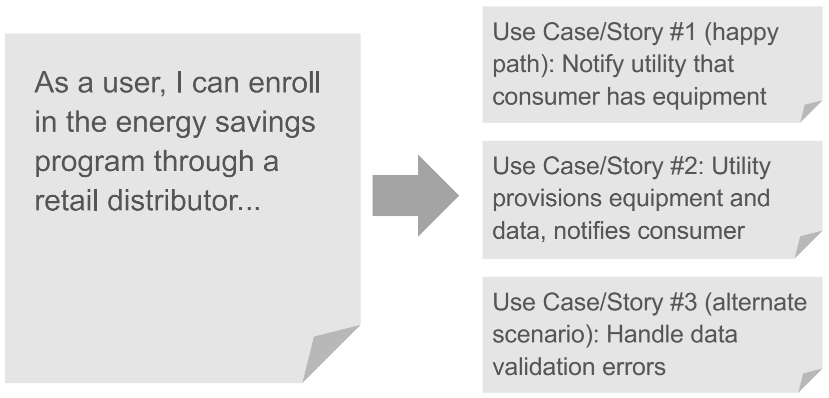
... I can edit my account settings

... I can cancel my account

… I can add more devices to my account

9. Split by use case scenarios

If use cases are used to represent complex interaction, the Story can be split via the individual scenarios.



As a user, I can enroll in the energy savings program through a retail distributor...

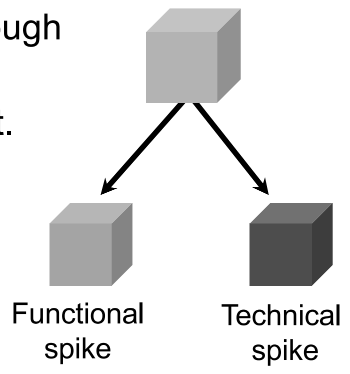
Use Case/Story #1 (happy path): Notify utility that consumer has equipment

Use Case/Story #2: Utility provisions equipment and data, notifies consumer

Use Case/Story #3 (alternate scenario): Handle data validation errors

10. Break out a spike

- A Story or Feature may not be understood well enough to estimate. Build a technical or functional spike to figure it out, then split the Story based on that result.
- Sometimes the team needs to develop a design, or prototype an idea
- Spikes are demonstrable, like any other Story


Functional spike
Technical spike