Boundaries

Topics

— Context

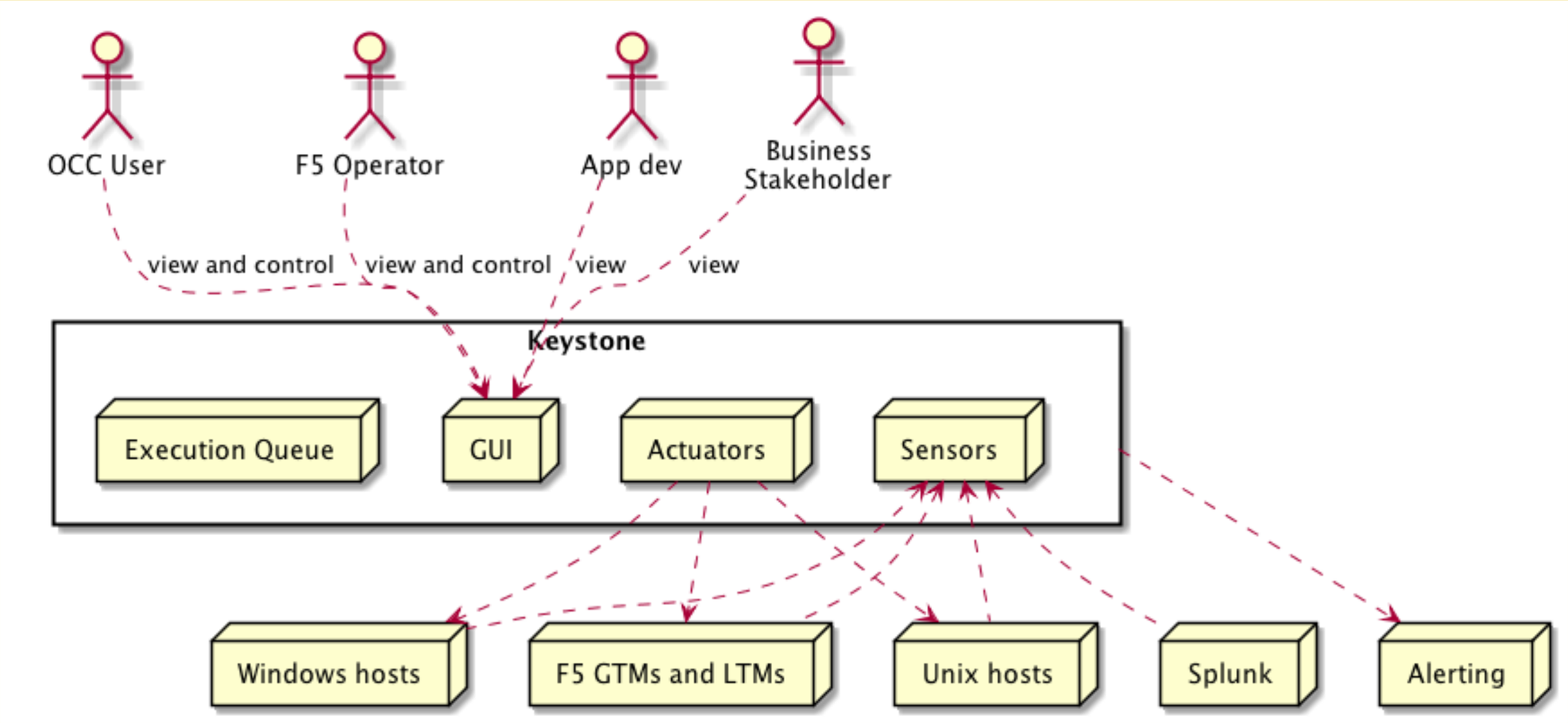
— System boundary

— Internal boundaries

Context

- Environment around your system
  - People grouped by role
  - Other systems

System Context Diagram



For each role

- It is a constituency
  - Find their needs
  - Use cases / stories
  - Determine how their needs get prioritized.

For each arrow across the boundary

- It is an interface
- For humans
  - UI Mockups / wireframes
- For APIs
  - Interface specs
  - Test harness
  - Mock

Interface Table

ID	Name	Initiation	Sync/Async	Frequency	Volume	Transport	Encoding	Semantics
IF1	F5 Commands	Push	Sync	1 / min	5 KB	HTTP	XML	F5 Big IP
IF2	Healthcheck	Push	Sync	1 / sec	1 KB	HTTP	HTTP	Request
IF3								
IF4								
...								

Other Attributes on an Interface

— Business criticality

— Security approach

— Failure handling (timeout, retry, fallback)

Internal boundaries

— Constrained by external interfaces

— Where architecture patterns appear

— Large impact on team structure and velocity

Creating Internal Boundaries

Improve Cohesion

- Bounded context
- Anticorruption layer

Reduce Coupling

- Align team structure to system structure
- Minimize crossing dependencies

David Parnas

"On the Criteria To Be Used in Decomposing Systems into Modules", CACM, 1972

KWIC Index

"Permuted Index"

Input: ordered set of lines, made up of ordered words, made up of ordered characters.
Output: Listing of all "circular shifts" of all lines, in alphabetic order

Modularization I

1. Input: read data lines, store them in core. Characters packed 4 per word. EOW special character.
2. Circular shift: prepare index; addr of first char of each shift, original index of line in array from module 1.
3. Alphabetizing: Take arrays from 1 & 2, produce same shape array as 2, but in alpha order.
4. Output: Using arrays from 1 & 3, format output
5. Control: sequence operations of 1 - 4, allocate memory, print errors.

Modularization II

1. Line storage: functional interface; functions to set/get char in line, get words in a line, delete words or lines
2. Input: read input call line storage.
3. Circular shifter: present same interface as 1, but appear to have all shifts of all lines
4. Alphabetizer: sort function ("ALPH") and access function ("iTH")
5. Output: Print output by calling "iTH" on the alphabetizer
6. Control: similar to first modularization

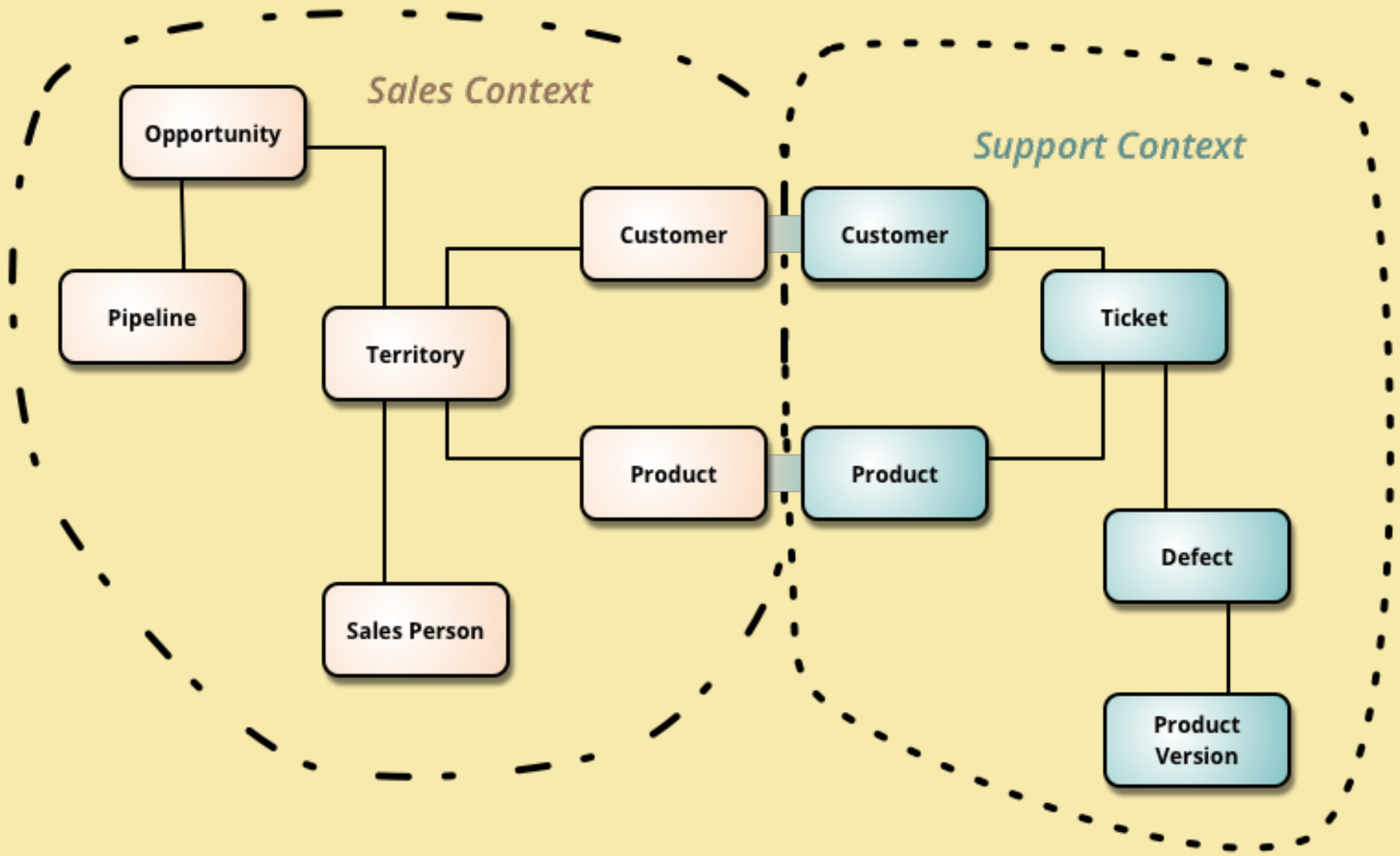
Impact of Changes?

1. Alter the input format.
2. Use offline storage for large jobs.
3. Stop using packed characters, write character per word.
4. Write index for circular shifts to offline storage instead of core.
5. Support Unicode

Hiding

Information, Decisions, Concepts

Bounded Contexts and Anticorruption Layers



Implications of Bounded Contexts

- Data duplication is OK
- Similar entity's lifecycles may differ
- Direct access to interior is prohibited

Tell, Don't Ask

— Send message, event

— Don't ask for data and make decisions

— Push decision to the holder of data

Stability Gradient

— Things close to the user change quickly.

— Things with high fan-in must change slowly.

— Apply different degree of governance.

Example: Platform vs. Tenant

From Joel Crabb's session "Platform architecture for omnichannel retail" from O'Reilly SA Con 2017.

— Tenants: Light governance, but strict rules about allowed coupling.

— Platform: More scrutiny to manage risk and improve long-term value