

Foundations of Software Engineering

Lecture 7: User stories and Risk
Michael Hilton

Announcements

- Time tracking survey (First September 23)
- Interview Quiz (1 Question, Due Wednesday at 11:59pm)

Examples adapted arbitrarily from prior years without identifying information!

REFLECTIONS ON REFLECTIONS

Reflection documents

Shallow

- Recite facts about what happened without adding anything.
- Recite statements from class without connecting to experience.
- State lessons learned without any reason why.

Good

- Extrapolate from the facts to add insight.
- Meaningfully connect prior experience or class material to assignment experience.
- Support lessons learned with evidence.

Shallow reflection examples

[PROCESS]

At our first meeting, we developed an initial outline of our approach. This was followed by preparing a list of tasks which were required for implementing the X system. Next, we divided the tasks among ourselves and came up with a rough timeline of the process to be followed.”

[SCHEDULE]

“Although we managed to meet all the milestones and implement all the desired features, the exact dates for the same could not be followed towards the end.”

[PLANNING]

“Learning how to use API X took a little longer than expected, which caused a setback of a day; but overall we managed to complete the entire project before the deadline and adhered to the timeline.”

[TEAM WORK / COMMUNICATION]

“We all agreed to use tool Y to keep in touch. We used it to announce when we started or completed individual tasks, current milestone statuses.. We also used Y to schedule a group meeting for the integration portion of our coding assignment”

Good reflection examples

[PLANNING / PROCESS]

“Since I was interested in the planning, we decided as a team I would be in charge of documenting our progress.. It worked really well to have one person managing what needed to get done or who needed to do it, and ensuring a shared single vision and set of goals as a group. However, there exist negatives approaching things this way...I found that my teammates sometimes would rely on me too heavily.”

[TEAM WORK / COMMUNICATION]

“An example of something that [would] work well is...issue tracking – something I asked them to do since first meeting. It’s easy to forget this information over time... If we had simply reminded ourselves on a regular basis, we would have had fewer problems forgetting these things.”

[PLANNING]

“People seemed to be annoyed because X “was not doing any work”. I believe X did the least amount of work, but we also assigned X the least amount of work. I wonder if this can all be traced back to the fact that X could not attend our first group meeting”

More good examples

[TEAMWORK]

“It helps to say ‘thank you’ before complaining about a teammate’s work. Only take conflict-inducing action if you think it is extremely important and are willing to follow up. Otherwise, you are wasting everyone’s time. Would we have treated each other differently if we had known we would be partnered up on more than just this assignment for the class?”

[TEAMWORK]

“two takeaways I had from this project are :

- It is best to present yourself as someone who is willing to help out, and do more than what was originally asked of you. This way, if people decline your offer to help out, they will be okay with the fact that you may not be working as hard as them at that point in time.
- Respect other people’s time and work, and take that into consideration when you decide to criticize their work or bring up issues. “

Also

- The homework document includes bulleted lists and prose outlining what a “good solution” looks like.
- Consider checking your submission against it, at the very least before submitting, if not sooner.

Learning goals

- Document requirements as user stories
- Evaluate the quality of a user story
- Understand risk and its role in requirements, specifically how it can be identified, analyzed, and then mitigated/handled in system design.

Requirements should be


- | | |
|----------------|---|
| 1. Correct | According to both the engineer and the customer |
| 2. Consistent | In that there are no conflicting requirements. Quality requirements are particularly dangerous. |
| 3. Unambiguous | Ambiguous: multiple readers can walk away with different but valid interpretations. |
| 4. Complete | Covers all required behavior and output for all inputs under all constraints. |
| 5. Feasible | Can it be done at all? Again, quality/non-functional reqs are particularly vulnerable. |
| 6. Relevant | Acceptance tests and metrics are possible/obvious. |
| 7. Testable | Organized, uniquely labeled. |
| 8. Traceable | |

Requirements Evaluation

Checklist Type		Source
Requirements		[Ackerman 1989], [Basili 1998], [Dunn 1984], [Freedman 1982], [Hollocker 1990], [Humphrey 1989], [Johnson 1995], [McConnell 1993], [NASA 1993], [Porter 1995], [SPAWAR 1997]
Design		[Basili 1998], [Dunn 1984], [Fagan 1976], [Freedman 1982], [Hollocker 1990], [Humphrey 1989], [Humphrey 1995], [Johnson 1995], [Kohli 1975], [Kohli 1976], [Maguire 1993], [McConnell 1993], [NASA 1993], [SPAWAR 1997]
Code	General	[Dunn 1984], [Fagan 1976], [Freedman 1982], [Hollocker 1990], [Humphrey 1989], [Jackson 1994], [Johnson 1995], [Kohli 1976], [McConnell 1993], [Myers 1979]
	Ada	[Humphrey 1997], [SPAWAR 1997]
	Assembler	[Ascoly 1976]
	C, C++	[Baldwin 1992], [Dichter 1992], [Humphrey 1995], [Maguire 1993], [Marick 1995], [NASA 1993], [SPAWAR 1997]
	Cobol	[Ascoly 1976]
	Fortran	[Ascoly 1976], [NASA 1993], [SPAWAR 1997]
	PL/I	[Ascoly 1976]
Testing		[Basili 1998], [Hollocker 1990], [Johnson 1995], [Larson 1975], [Maguire 1993], [McConnell 1993], [NASA 1993], [SPAWAR 1997]
Documentation		[Freedman 1982], [Hollocker 1990], [Humphrey 1989], [SPAWAR 1997]
Process		[Freedman 1982], [McConnell 1993], [SPAWAR 1997]

Table 1. Inspection checklists by type

Requirements Evaluation


 **github** / **octo-repo** Private


Watch 5


Star 0


Fork 0


<> Code


 **Issues** 5


 Pull requests 1

 Projects 1

 Wiki

 Pulse

 Graphs


 Settings

Filters ▾

Labels

Milestones

New issue

☐  **5 Open** ✓ 0 Closed


Author ▾


Labels ▾


Milestones ▾


Assignee ▾


Sort ▾

☐  **Document new features**
#6 opened 20 minutes ago by jleaver

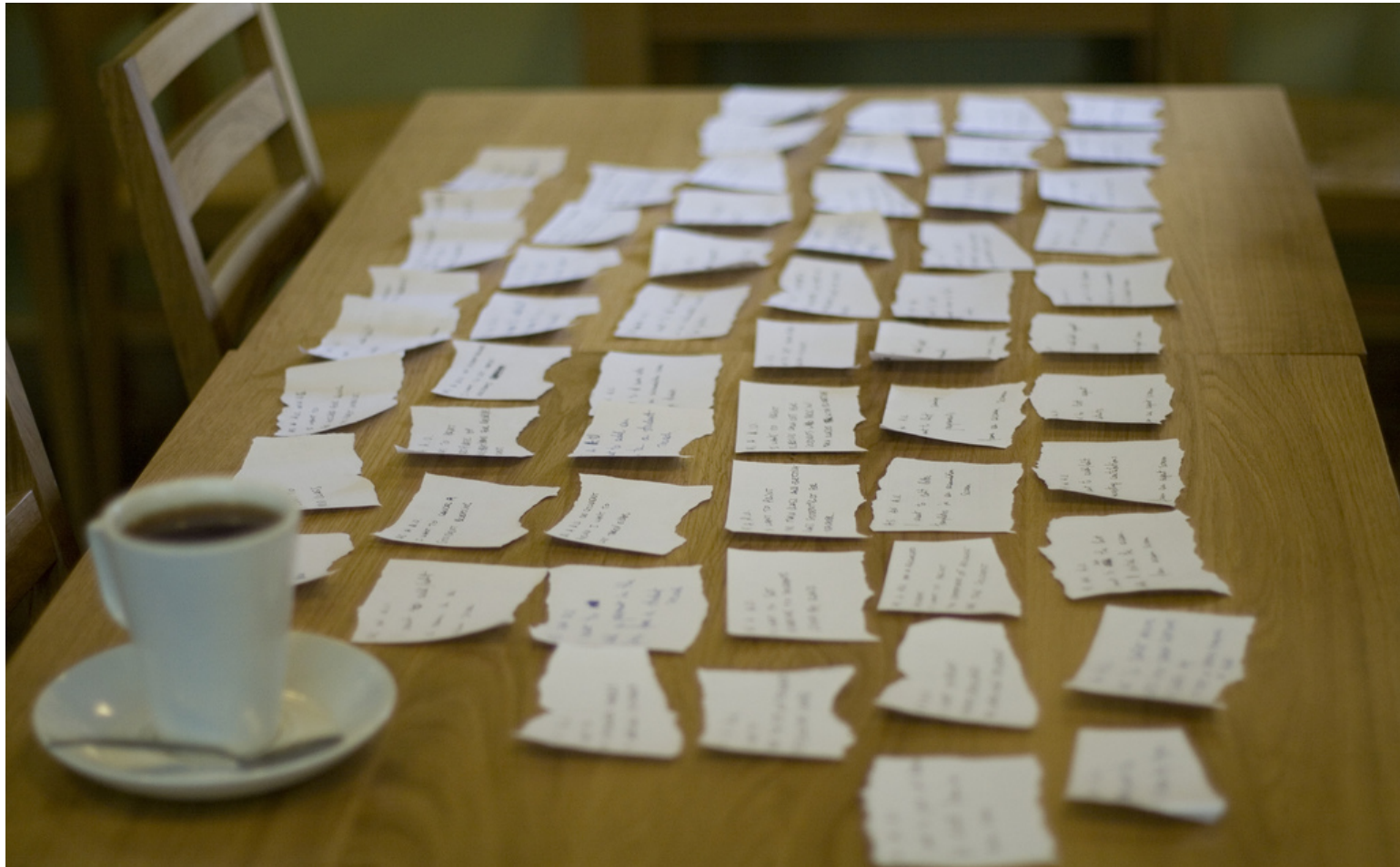
☐  **Update cat images**
#5 opened 21 minutes ago by jleaver

☐  **Fix failing tests**
#4 opened 21 minutes ago by jleaver

☐  **Add section for new features**
#3 opened 25 minutes ago by jleaver

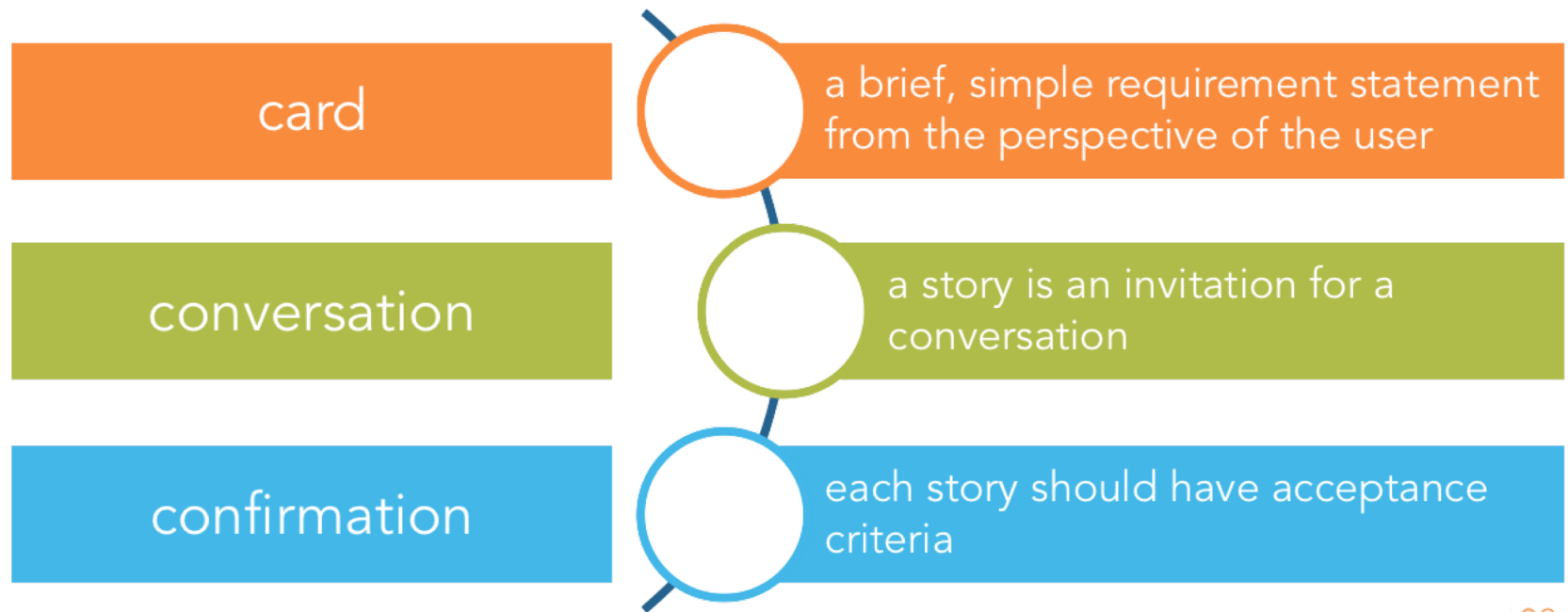
☐  **Create a README**
#1 opened 11 days ago by emilyistoofunky

User Stories



Source: <https://www.flickr.com/photos/jakuza/2728096478>

User Stories



one 80
services

The card

- “As a [role], I want [function], so that [value]”
- Should fit on a 3x5 card

The conversation

- An open dialog between everyone working on the project and the client
- Split up Epic Stories if needed

The Confirmation

- A confirmation criteria that will show when the task is completed
- Could be automated or manual

Exercise



<https://www.bird.co/>

How to evaluate user story?

Follow the INVEST
guidelines for good
user stories!



one | 80
services



Independent



- Schedule in any order.
- Not overlapping in concept
- Not always possible

Negotiable



- Details to be negotiated during development
- Good Story captures the essence, not the details

Valuable



- This story needs to have value to someone (hopefully the customer)
- Especially relevant to splitting up issues

Estimable



- Helps keep the size small
- Ensure we negotiated correctly
- “Plans are nothing, planning is everything” -Dwight D. Eisenhower

Small



- Fit on 3x5 card
- At most two person-weeks of work
- Too big == unable to estimate

Testable



- Ensures understanding of task
- We know when we can mark task “Done”
- Unable to test == do not understand

Activity

Follow the INVEST
guidelines for good
user stories!



one | 80
SERVICES



Risk



Tony Webster ✓
@webster

Follow

▼

I appreciate the honesty.

Pick a password

Don't reuse your bank password, we didn't spend a lot on security for this app.

At least 6 characters

your password

Continue

8:20 PM - 15 Sep 2018

5,868 Retweets 15,672 Likes



58

5.9K

16K

What are risks?

- A **risk** is an uncertain factor that may result in a loss of satisfaction of a corresponding objective

For example...

- System delivers a radiation overdose to patients (Therac-25, Theratron-780)
- Medication administration record (MAR) knockout
- Premier Election Solutions vote-dropping “glitch”

How to assess the level of risk?

- Risks consist of multiple parts:
 - Likelihood of failure
 - Negative consequences or impact of failure
 - Causal agent and weakness (in advanced models)
- Risk = Likelihood x Impact

CVSS V2.10 Scoring

The Common Vulnerability Scoring System consists of:

- 6 base metrics (access vector, complexity, confidentiality impact, ...)
- 3 temporal metrics (exploitability, remediation, ...)
- 5 environmental metrics; all qualitative ratings (collateral damage, ...)

BaseScore =

$\text{round_to_1_decimal}(((0.6 * \text{Impact}) + (0.4 * \text{Exploitability}) - 1.5) * f(\text{Impact}))$

Impact =

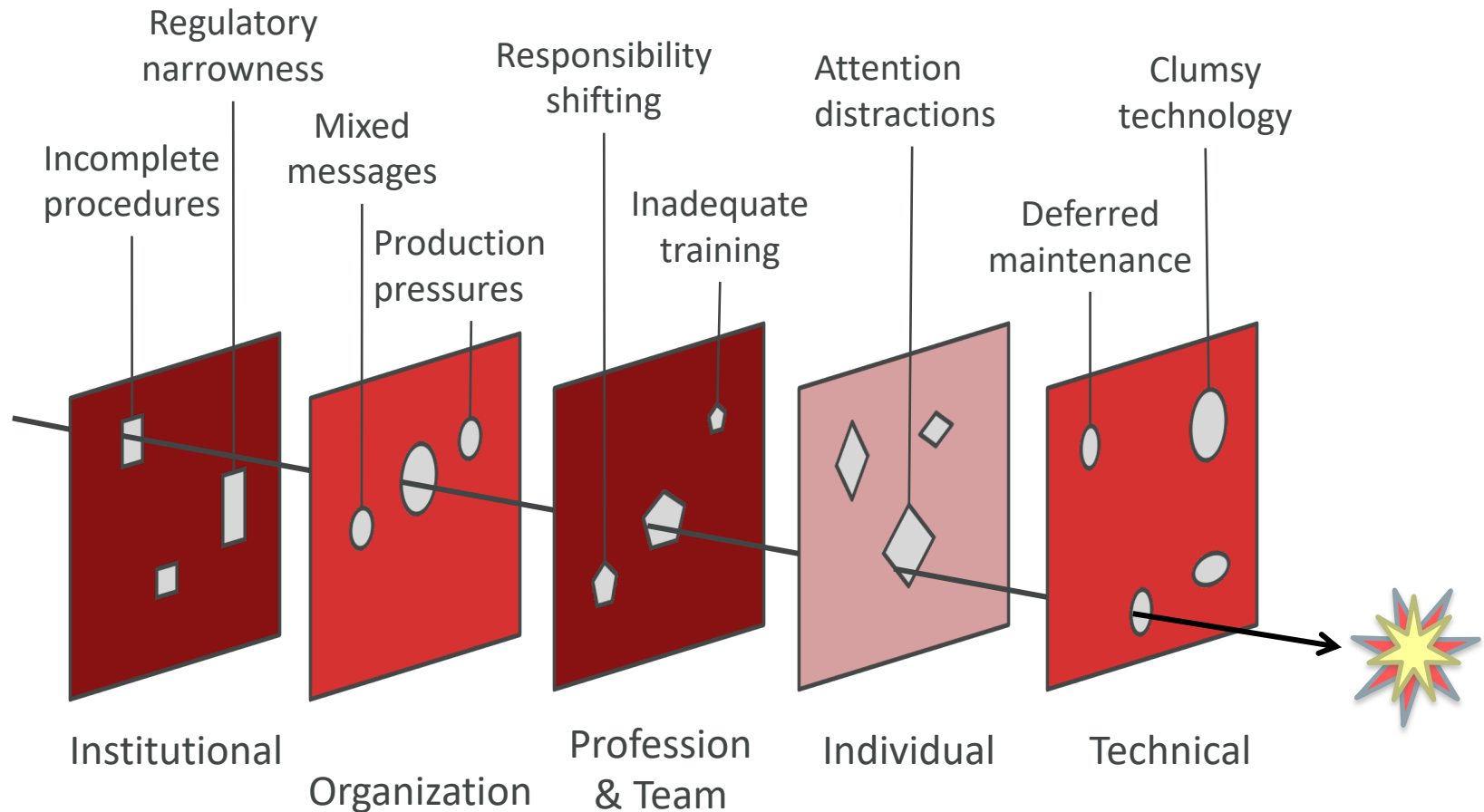
$10.41 * (1 - (1 - \text{ConfImpact}) * (1 - \text{IntegImpact}) * (1 - \text{AvailImpact}))$

Exploitability =

$20 * \text{AccessVector} * \text{AccessComplexity} * \text{Authentication}$

f(impact) = 0 if Impact=0, 1.176 otherwise

The Swiss cheese model



Modified from Reason, 1999, by R.I. Crook

Aviation failure impact categories

- **No effect** – failure has no impact on safety, aircraft operation, or crew workload
- **Minor** – failure is noticeable, causing passenger inconvenience or flight plan change
- **Major** – failure is significant, causing passenger discomfort and slight workload increase
- **Hazardous** – high workload, serious or fatal injuries
- **Catastrophic** – loss of critical function to safely fly and land

Risk assessment matrix



- MIL-STD-882E

<https://www.system-safety.org/Documents/MIL-STD-882E.pdf>

TABLE III. Risk assessment matrix

RISK ASSESSMENT MATRIX				
SEVERITY PROBABILITY	Catastrophic (1)	Critical (2)	Marginal (3)	Negligible (4)
Frequent (A)	High	High	Serious	Medium
Probable (B)	High	High	Serious	Medium
Occasional (C)	High	Serious	Medium	Low
Remote (D)	Serious	Medium	Medium	Low
Improbable (E)	Medium	Medium	Medium	Low
Eliminated (F)	Eliminated			

DECIDE Model



Detect that the action necessary

Estimate the significance of the action

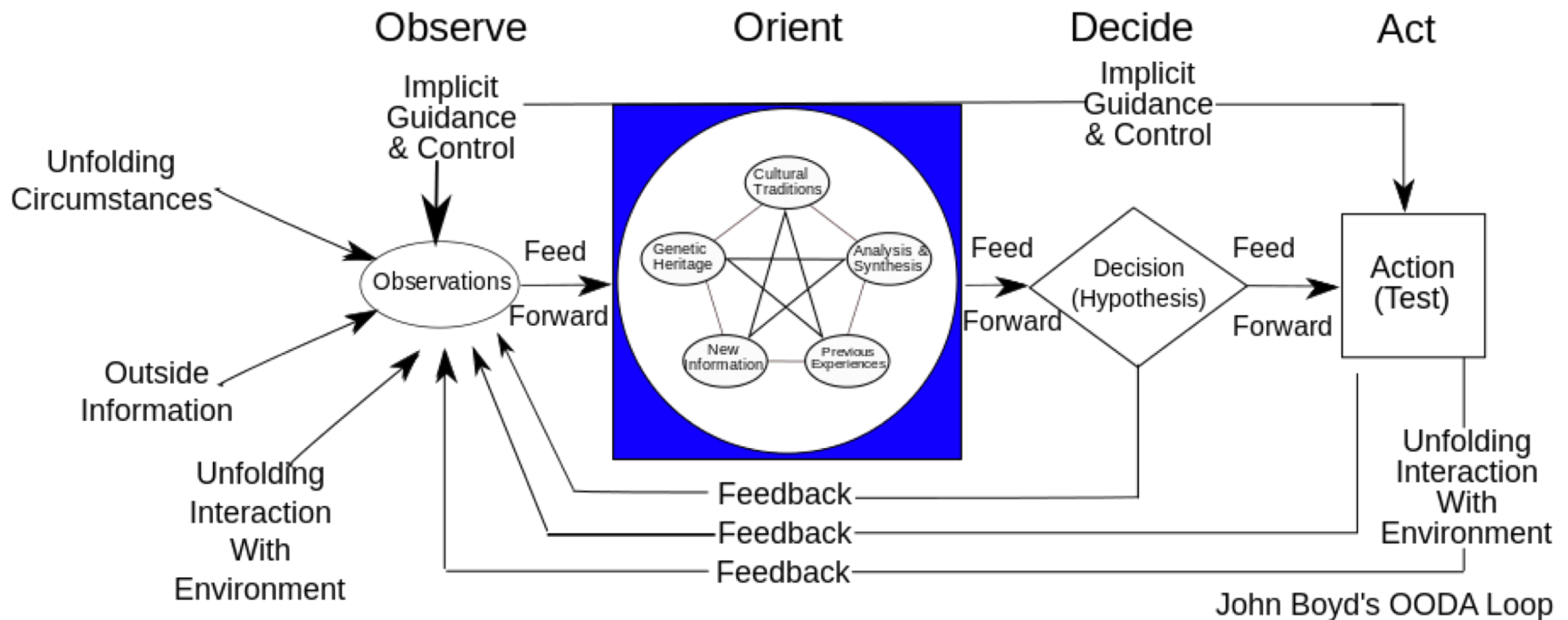
Choose a desirable outcome

Identify actions needed in order to achieve
the chosen option

Do the necessary action to achieve change

Evaluate the effects of the action

OODA Loop



Bird Risks

