CS-639 — Interaction Design Studio

Proactivity & Timing Studio | In-Class Exercise

Professor Bilge Mutlu

Today's Exercise

Design the same proactive feature at three initiative levels — Act, Ask, and Wait — then evaluate each with the four designer questions and design the threshold logic.

— **ACT:** System executes autonomously; user sees the result

— **ASK:** System presents a suggestion; user decides

— **WAIT:** System gathers information silently; user is unaware

The question isn't "which level is best?" — it's "what determines when the system should shift between them?"

Step 1: Choose Your Domain and Proactive Feature

Pick a domain — your A1 app, a class exercise, or a common app (email, music, navigation, health, shopping).

Pick one proactive feature — something the system does (or could do) without being explicitly asked. Good candidates:

— Suggesting content, actions, or responses
— Rerouting, rescheduling, or reorganizing
— Alerting, reminding, or warning
— Auto-completing, auto-saving, or auto-correcting

Identify: What is the system inferring? What action does it take? Who benefits?

Step 2: Design at Three Initiative Levels

Sketch the same proactive feature three times — each at a different initiative level.

ACT
System executes without asking. User sees the result — or doesn't.

What does the user see after the system acts? How do they know it happened? Can they undo it?

ASK
System presents a suggestion. User approves, modifies, or dismisses.

How is the suggestion presented? What information does the user need to decide? What happens if they ignore it?

WAIT
System gathers information silently. User is unaware.

What is the system observing? What would make it confident enough to shift to Ask or Act?

Step 3: Evaluate with the Four Designer Questions

For each of your three designs, answer the four designer questions from Monday's lecture:

Question	ACT	ASK	WAIT
1. Is it worth acting?			
2. Do I know enough?			
3. Is the timing right?			
4. Can the user stay in control?			

Annotate each design: Where does it satisfy the questions? Where does it fail? Which question is hardest to answer "yes" for the ACT version?

Step 4: Design the Threshold

This is what's new vs. W08. Don't just design three versions — design the **logic that determines when the system shifts between them.**

For each transition, specify:

— **WAIT → ASK:** What signal or confidence level triggers a suggestion?
— **ASK → ACT:** What makes the system confident enough to act autonomously?
— **ACT → ASK:** What would cause the system to become less certain and fall back to asking?

Be specific. Not "when confidence is high" — what is the system measuring? What threshold? What context?

Example: "WAIT → ASK when the system detects a date/time pattern in an email with >70% confidence. ASK → ACT when the user has accepted 5+ similar suggestions. ACT → ASK when the meeting conflicts with an existing event."

What Good Looks Like

Strong submissions:

— Three designs that look and feel genuinely different — not the same screen with different labels

— Four-question evaluation that reveals real tradeoffs between levels

— Threshold logic that is specific and testable — concrete signals, not vague "confidence"

— Clear reasoning about which level is the default and why

Common pitfalls:

— Three designs that are the same UI with more/less confirmation

— Threshold logic that says "when confidence is high" without defining what that means

— Assuming ACT is always the goal — sometimes WAIT is the right default

— Ignoring what happens when the system transitions at the wrong moment

Example: Email Meeting Detection

ACT: System detects "Let's meet Tuesday at 2pm" in email, creates calendar event automatically. User sees notification: "Meeting added: Tuesday 2pm with Sarah."

ASK: System highlights date/time text, shows card: "Add to calendar? Tuesday 2pm — Sarah Chen — Conference Room B." User taps Confirm or Dismiss.

WAIT: System notes date/time patterns in email thread, enriches calendar view with "possible meetings" badge. User can tap to review.

Threshold logic:

- **WAIT → ASK:** Date/time + person detected, >70% confidence, no conflicting event
- **ASK → ACT:** User has accepted 5+ similar suggestions in the past month
- **ACT → ASK:** Conflicting event exists, or meeting involves 4+ people, or user previously dismissed a similar suggestion

Default: ASK — because meeting scheduling errors are moderately costly and hard to undo silently.

Get Started

1. **3 min:** Choose domain and proactive feature
2. **15 min:** Sketch at all three initiative levels (Act, Ask, Wait)
3. **10 min:** Evaluate each with the four designer questions — fill in the table
4. **5 min:** Design the threshold logic — specify transitions between levels
5. **Remaining:** Finalize annotations

Due Monday morning on Canvas — all three designs with four-question evaluation, threshold logic, and your recommended default level with justification.

Friday critique: "Are your three designs genuinely different? Is the threshold logic specific? What happens if the system transitions at the wrong moment?"