1. Two Kinds

A. All-or-Nothing

An action, it is often a key step in such a way that the key never

B. Atomic

A key action is always either to discover the way to implement

An action is for a higher-level (later)
y = 2x + 5
x = 3

Corrected expression:

If their effect from the point of view used to propagate command actions have the box propagate correctly and then together ensure consistency while aligning with their inverses as if the action had occurred completely before or after one another.
Put.

Write a block to the disk.

Overall system fault tolerance. Specification.

Error-free op.

Tolerated errors.

Untolerated errors.

Process, memory A as F LCD.

Memory, memory B as F LCD.

All following programs.

The system is fail-fast.

Detected errors: The system is fail-fast.

Initiating any further puts to the disk.

And restore from a clean disk before.

i.e. the h/l or os detection for failure.
Untolerated errors: something fails in the H/D or O/S, the processor muddles ahead and puts corrupted data to the disk before detecting the failure.

Disk storage fault model:
- Assume perfect disk model i.e. no disk decay
- Careful disk model:
  - error-free op: careful-get returns the result of the most recent call to careful-put at sector number on track, with status = O.K.
2. All or nothing - pull

1. Almost all or nothing - pull

corrupted before detection.
no more than one forget is
F:8. model of preoc, memory is os track

- Assume HR is a checksum
- On one another 9% chill
  careful - pull matching correspond does
  disk buffer in volatile storage, and
d  careful - put and correspond for

detectable error. This as triggers dummy
end procedure

else data \rightarrow data2' \quad \text{from data} \rightarrow \text{data1}' \quad f (\text{d1} = \text{d2}) \quad \text{then data} \rightarrow \text{data1}'

\begin{align*}
\text{CAFE} & \text{ FUR - GET} (\text{d1}, \text{sec}, \text{sec}) \\
\text{CAFE} & \text{ FUR - GET} (\text{sec2}, \text{sec}, \text{sec2}) \\
\text{CAFE} & \text{ FUR - GET} (\text{d2}, \text{sec}, \text{sec1}) \\
\text{CAFE} & \text{ FUR - GET} (\text{d3}, \text{sec}, \text{sec1})
\end{align*}

\underline{procedure \text{ ALMOST - ALL - OR - NOT - HIND - C} (\text{sec}, \text{sec}, \text{sec}, \text{sec})} \\
\underline{end procedure}

\begin{align*}
\text{CAFE} & \text{ FUR - PUT} (\text{d3}, \text{sec}, \text{sec3}) \\
\text{CAFE} & \text{ FUR - PUT} (\text{d1}, \text{sec}, \text{sec2}) \\
\text{CAFE} & \text{ FUR - PUT} (\text{d2}, \text{sec}, \text{sec1}) \\
\text{CAFE} & \text{ FUR - PUT} (\text{d4}, \text{sec}, \text{sec})
\end{align*}

\underline{procedure \text{ ALMOST - ALL - OR - NOT - HIND - PUT (data, sec), sec, sec, sec} } \\
\underline{end procedure}

\begin{align*}
\text{sec} & \quad \text{sec} \\
\text{sec} & \quad \text{sec} \\
\text{sec} & \quad \text{sec}
\end{align*}

\text{Virtual cell}

\text{Put}
Assumes that initially \( \text{sec}.51 = \text{sec}.52 = \text{sec}.53 \).
And Procedure

CAReFul - put (doh 1, sec 52); r
CAReFul - put (doh 2, sec 52); r
CAReFul - put (doh 2, sec 51); r
return.

if (doh 2 = doh 3) if 2 or 3 >

else 3 <-

if (doh 1 = doh 2) return;

if (doh 1 = doh 2) return;

CAReFul - get (doh 3, sec 45);
CAReFul - get (doh 2, sec 42);
CAReFul - get (doh 1, sec 41);

Procedure CHECK-AND-REPAIR (sec) // make sure

Procedure ALl-OR-NOTHING-Put (dah, sec)

ALl-OR-NOTHING-Put (dah, sec)

CHECK-AND-REPAIR (sec)
Data state

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>0</td>
<td>0</td>
<td>b</td>
<td>0</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>s2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>b</td>
<td>n</td>
</tr>
<tr>
<td>s3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>b</td>
</tr>
</tbody>
</table>

0: old  b: bad  n: new

Why only 7?

State space?: $3^3 = 27$

Claim is only 7 out of 27 states are possible

because only one sector can get corrupted.

either is fine
AoN PUT can be viewed as a special case/version of shadow copy technique.

Systematic AoN — Version histories for ops which modify arbitrary data structures.

"Delete Nothing"

Journal Storage:
1. Version histories for each var/record.
2. Uses shadow copy technique.