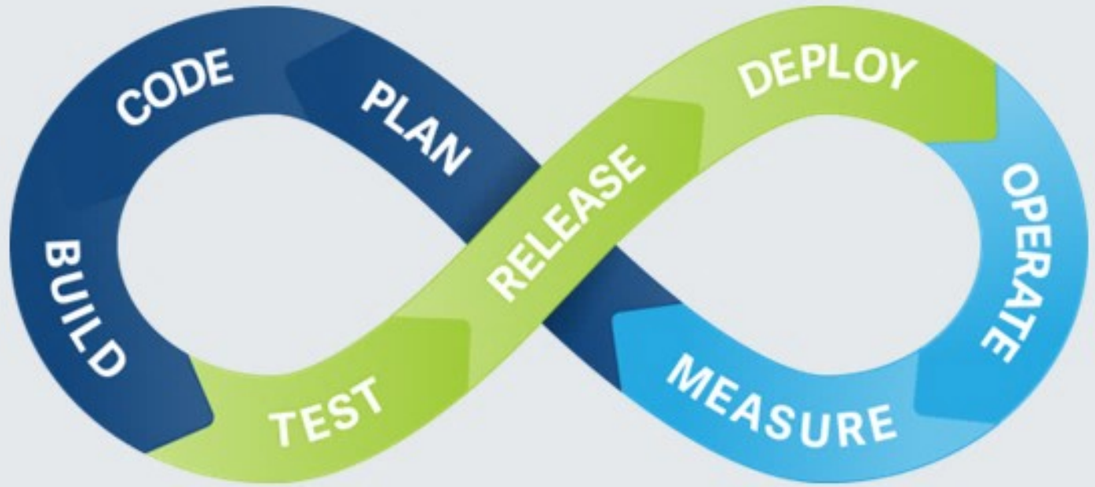
Towards Predicting the Impact of Software Changes on Building Activities

M. Tufano, H. Sajnani, K. Herzig

Continuous Integration



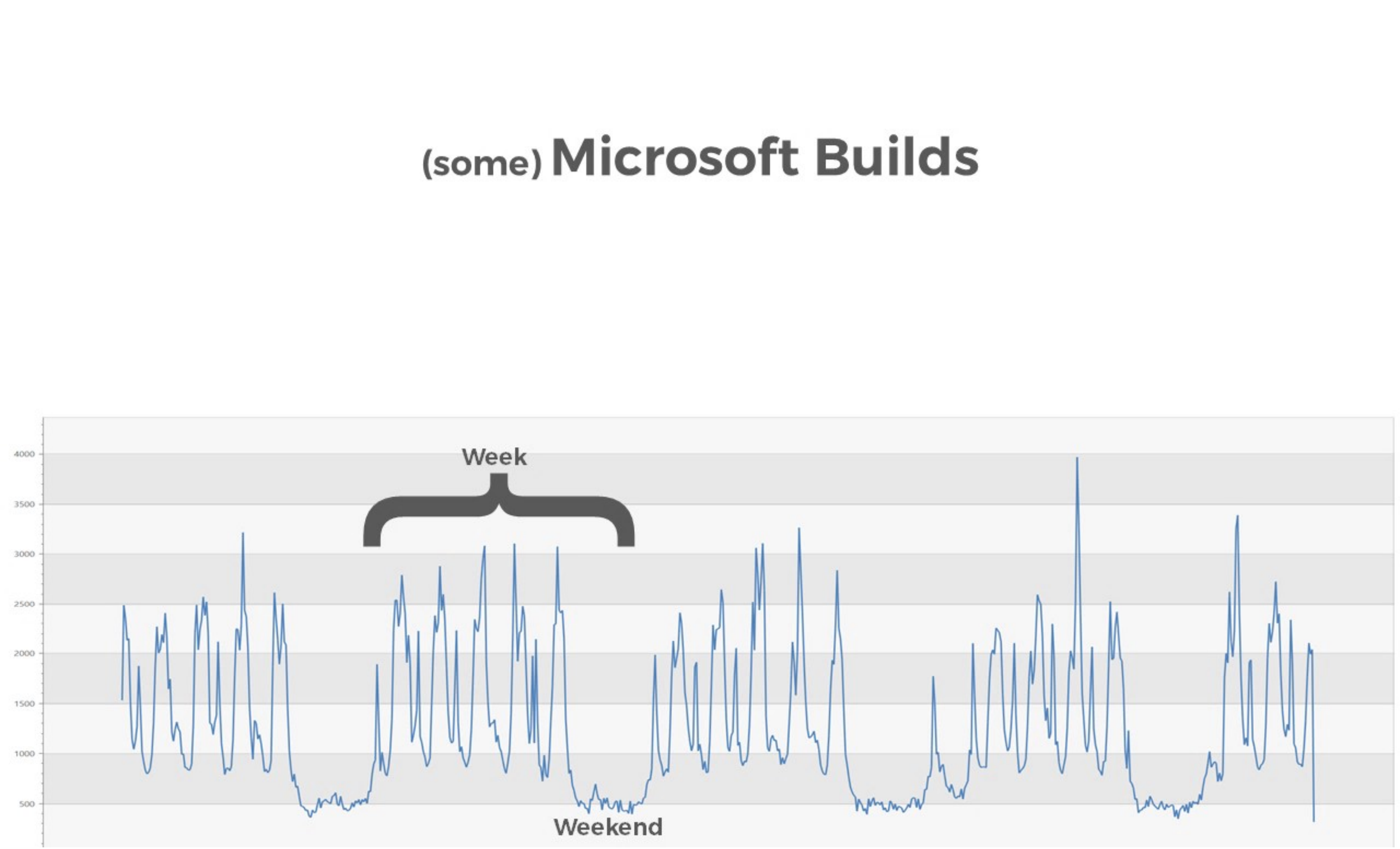
Continuous Integration

Developers build daily.

Continuous Integration

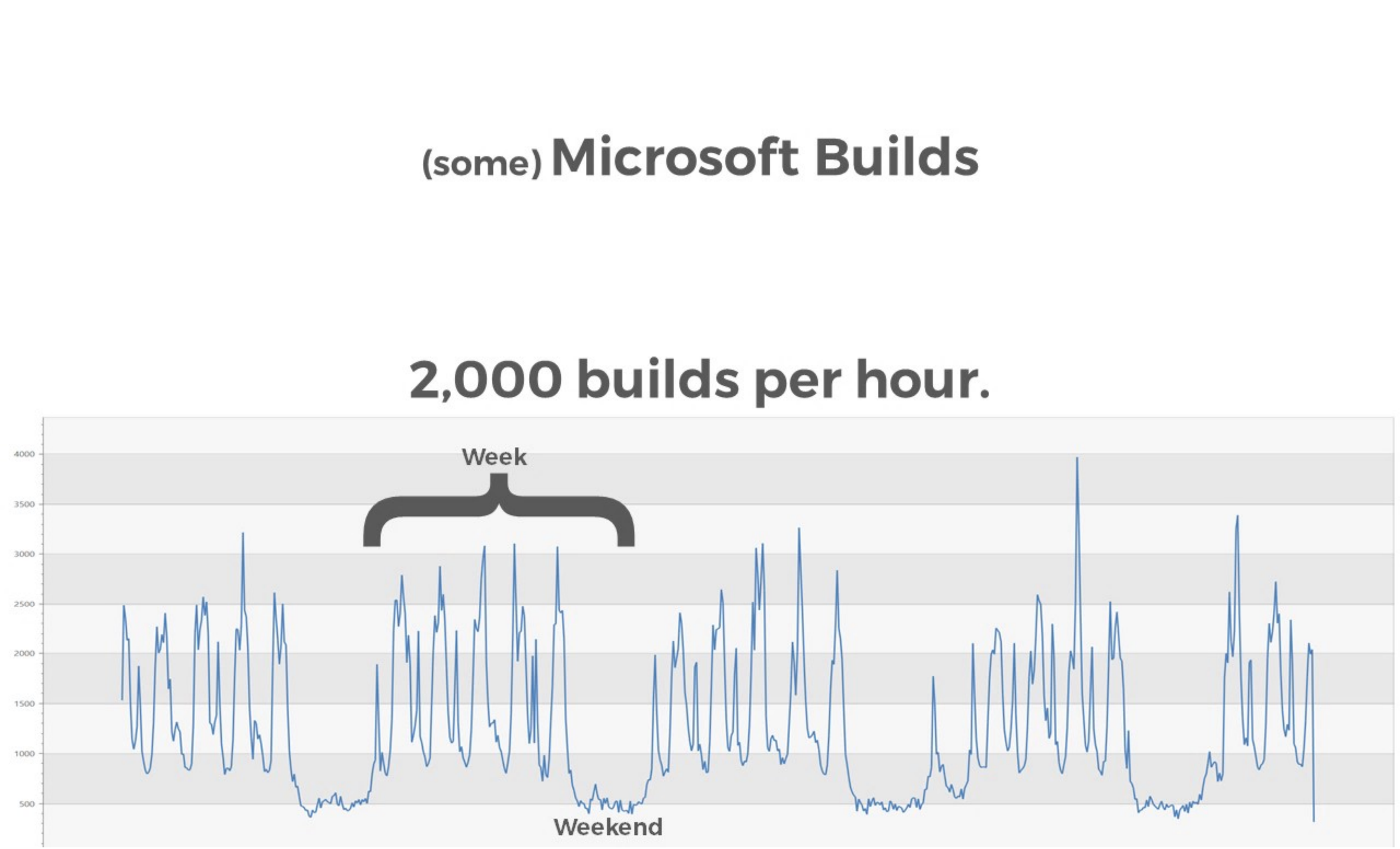
Developers build daily.
Many times a day.

(some) Microsoft Builds



(some) Microsoft Builds

2,000 builds per hour.



Builds in the Cloud 

Distributed and parallel builds on remote cloud infrastructures

Microsoft **CloudBuild**

Google **Bazel**

facebook **Buck**

CloudBuild

Distributed builds on many machines in the cloud

Parallelized build tasks

Content-based **cache** to accelerate builds

Builds, test, code analysis, drops, package, and storage.

Faster Builds 

Build time is the bottleneck for shipping faster.

Not only improvements on the infrastructure side.

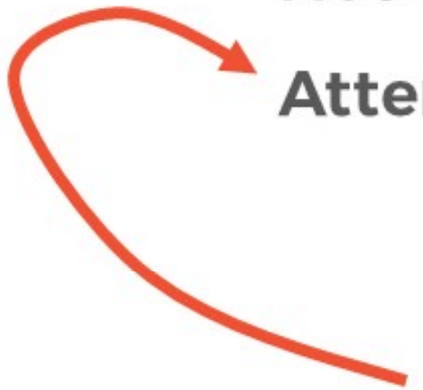
Attention to developers' changes.

Faster Builds 

Build time is the bottleneck for shipping faster.

Not only improvements on the infrastructure side.

Attention to developers' changes.

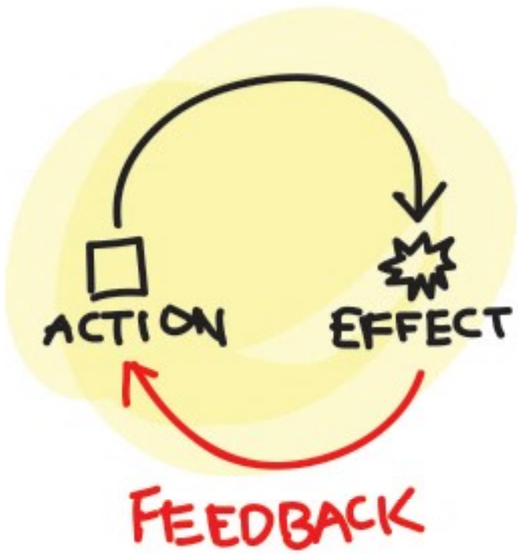


Focus of this paper!

Faster Builds

Developer Changes

- Dependencies
- Architectural

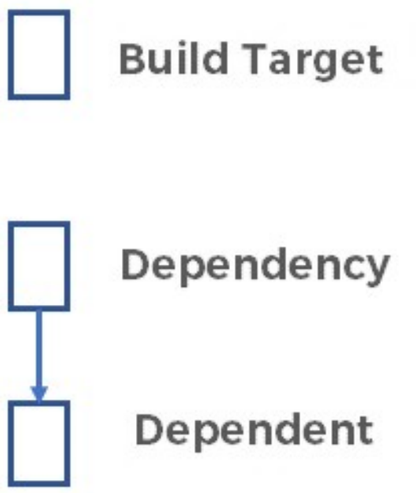


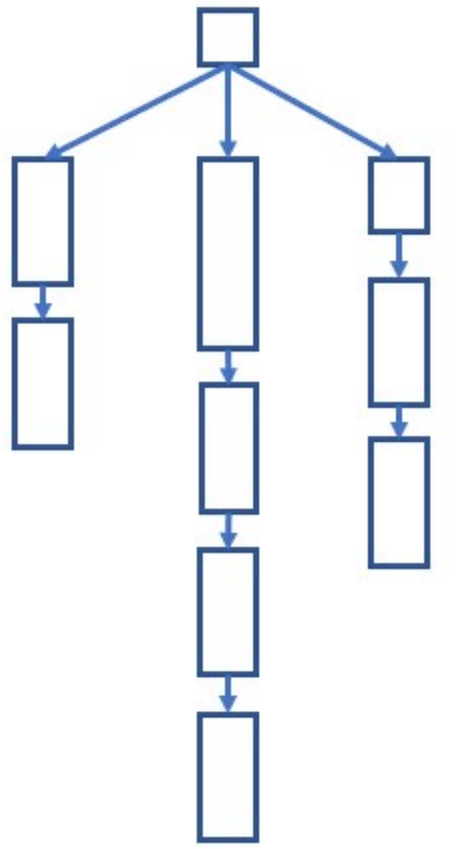
Early Feedback

- Awareness of impact
- Early restructuring
- Avoid build time regression

Predicting the impact is challenging with modern cached build systems.

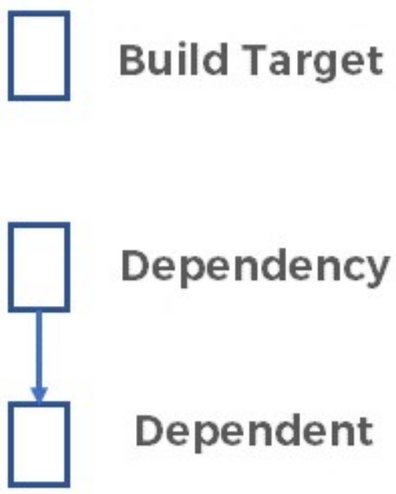
Build Dependency Graph

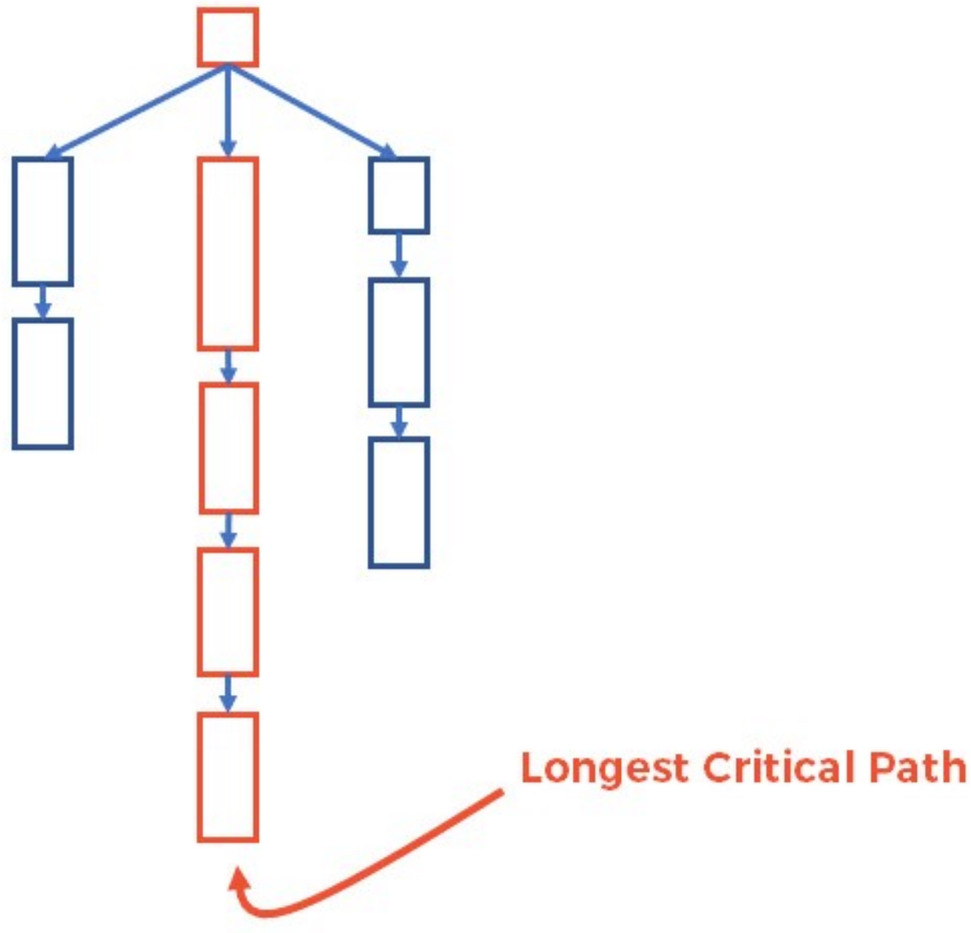




Build Dependency Graph

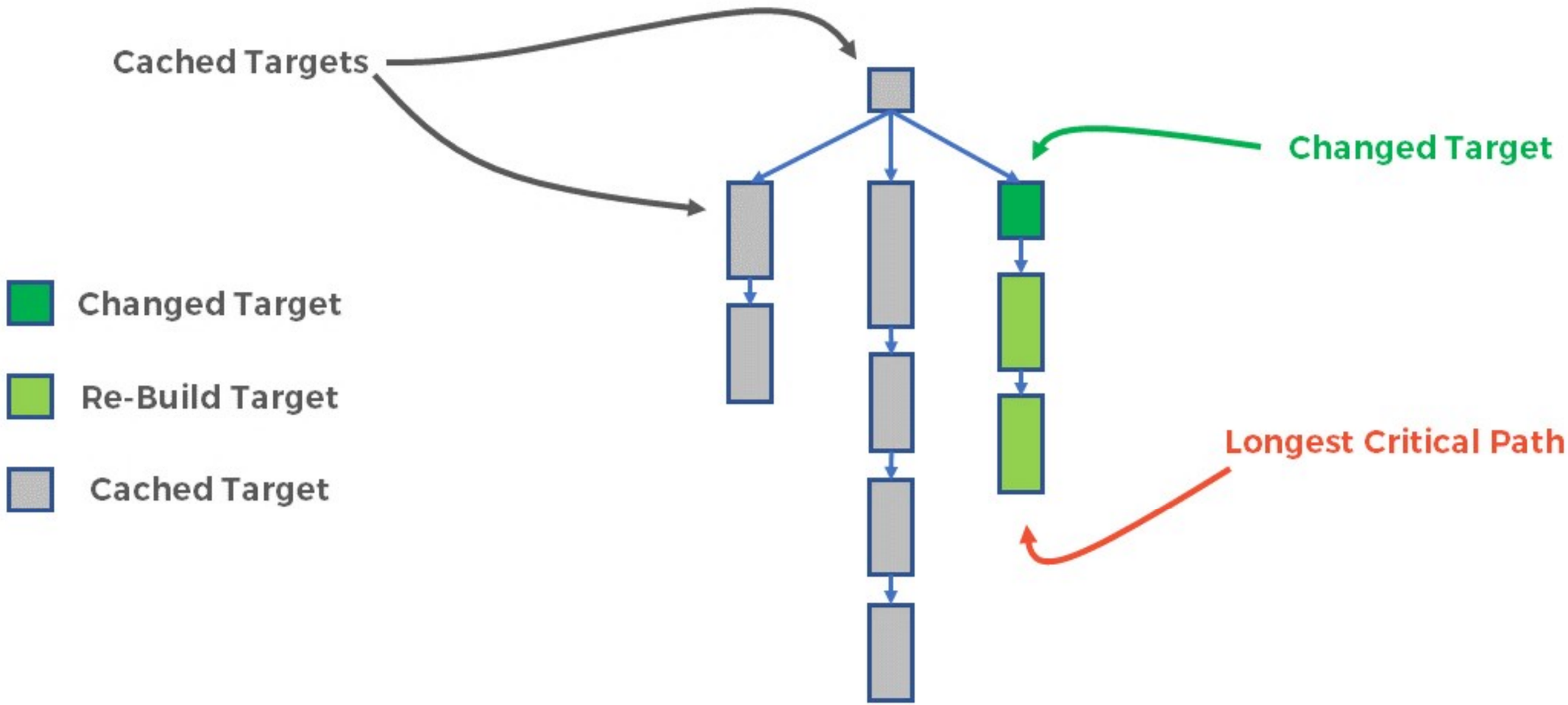
Static Full Build



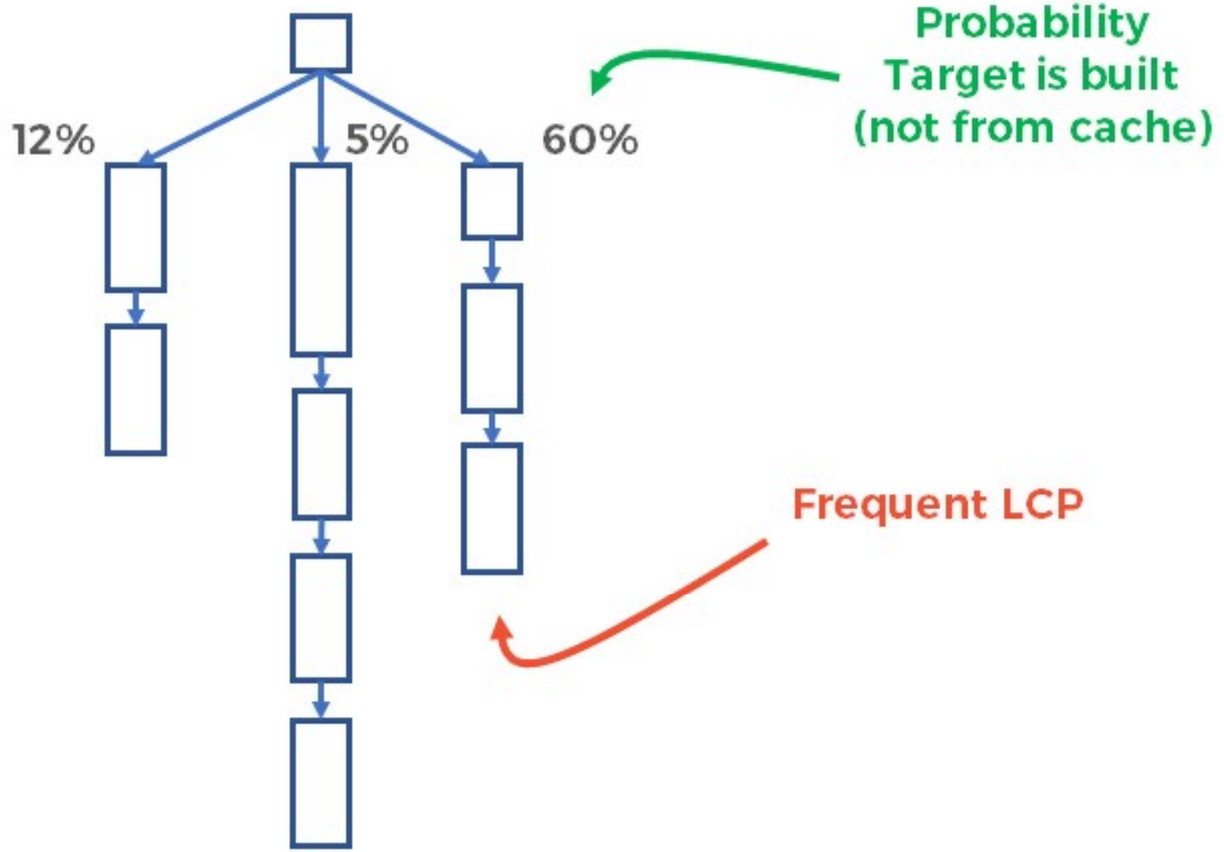


Cached Build System

Incremental Build

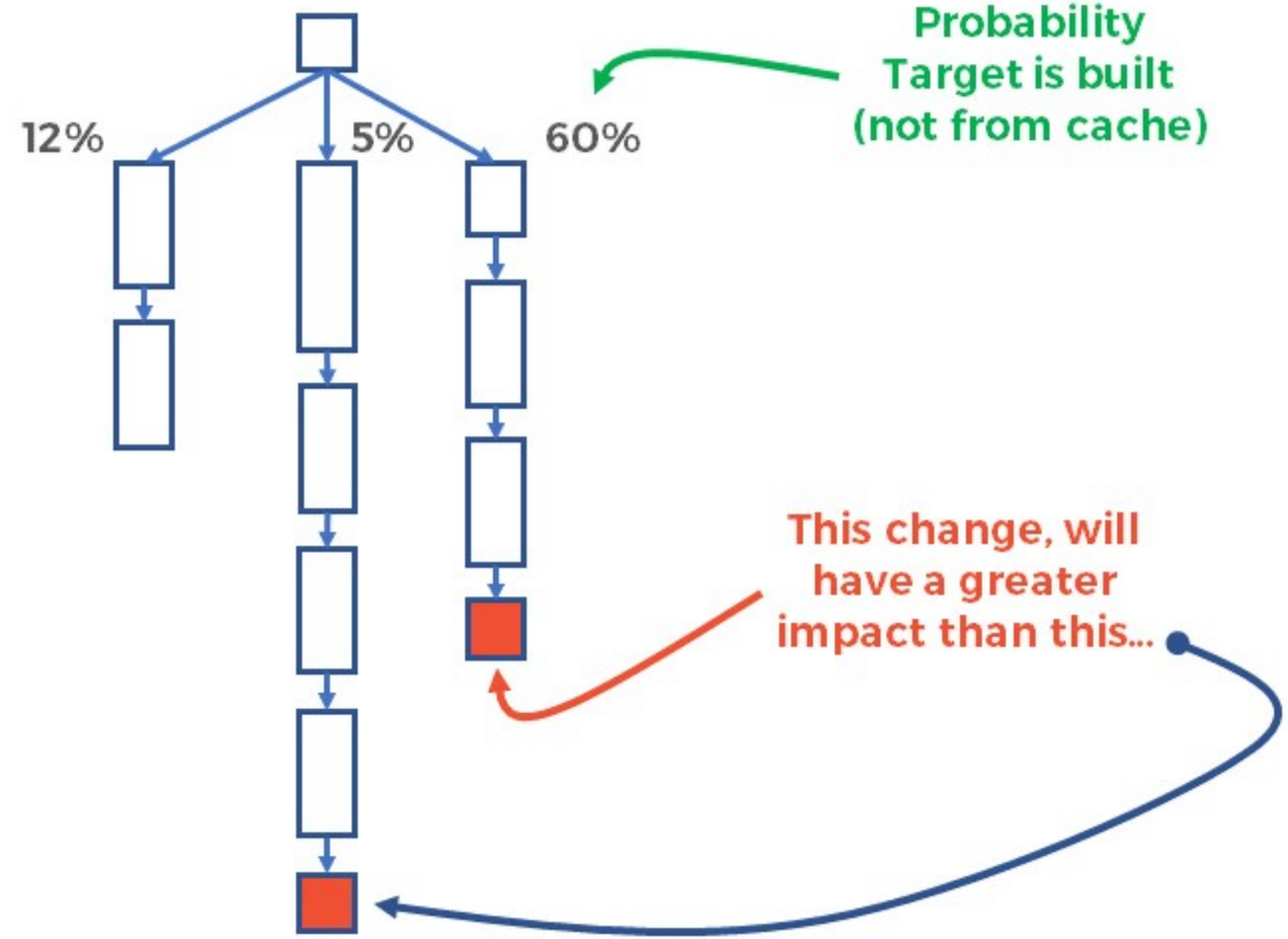


Cached Build System



Cached Build System

Added Target



Impact of Software Changes on Building Activities

Predict Impact

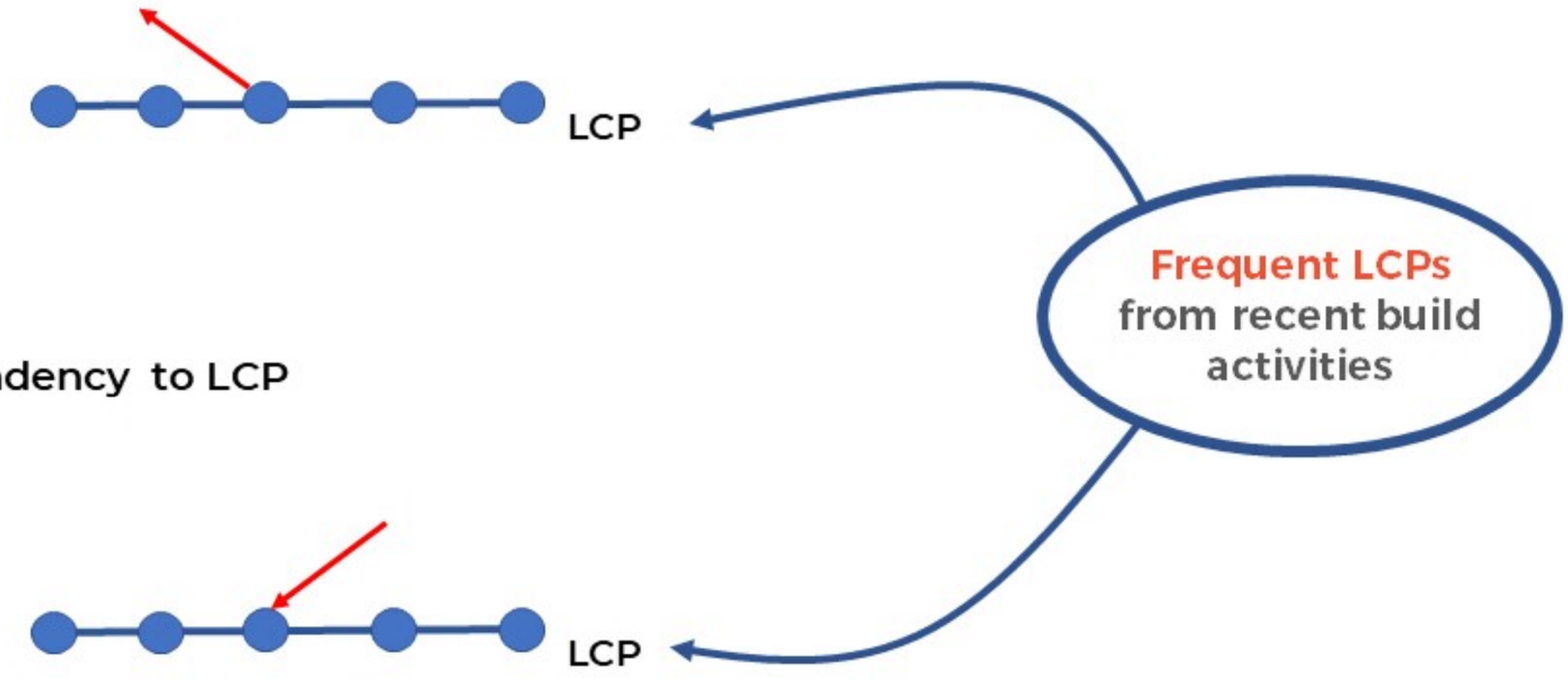
- Build Time increase
- Percentage of future builds affected

Approximation using Telemetry data

- Probabilities
- Target Execution Time

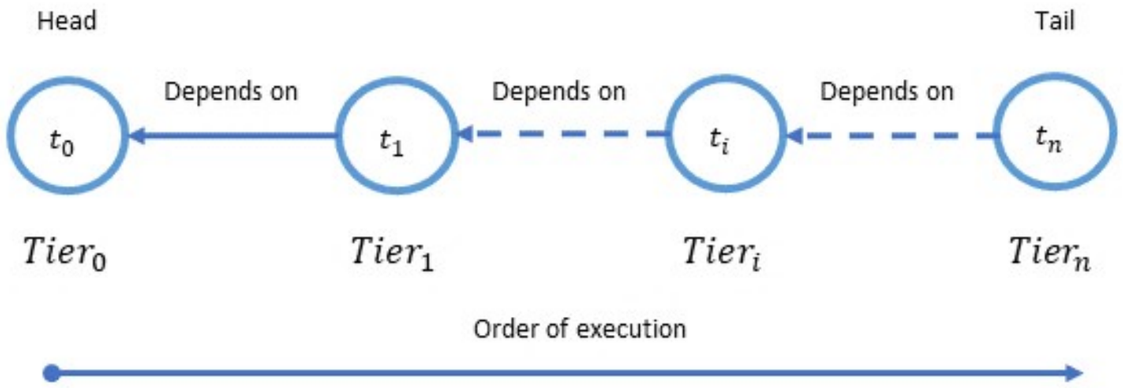
Software Changes introducing dependencies

- New Outward dependency from LCP



- New Inward dependency to LCP

Standards

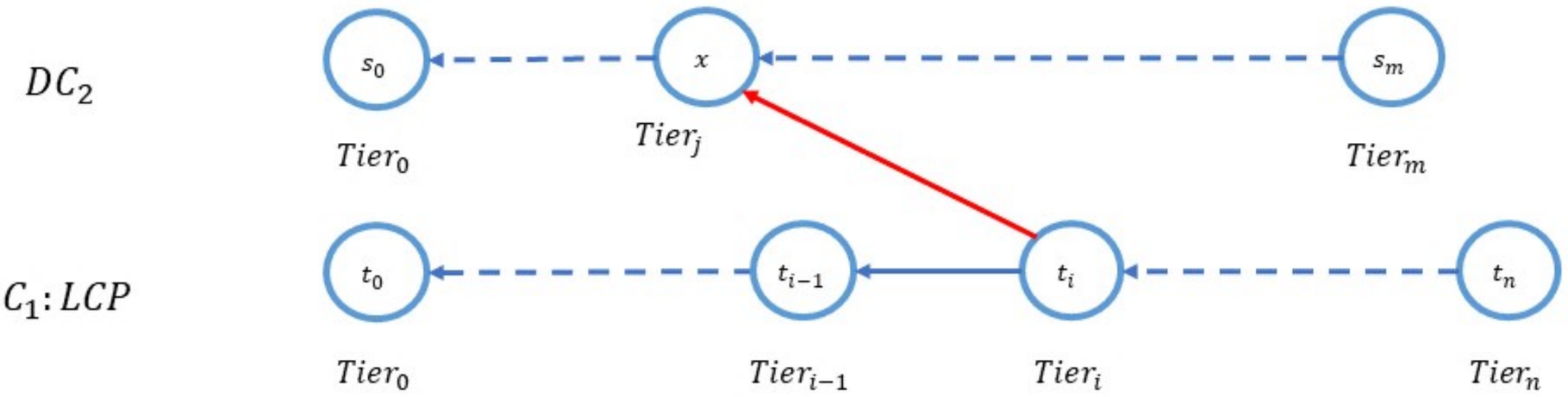


Functions

$ExecTime(t_0, \dots, t_i)$	Estimation of execution time of a sequence of targets
$BuildCoverage(t_i)$	Estimation of percentage of builds building the target (not from cache)

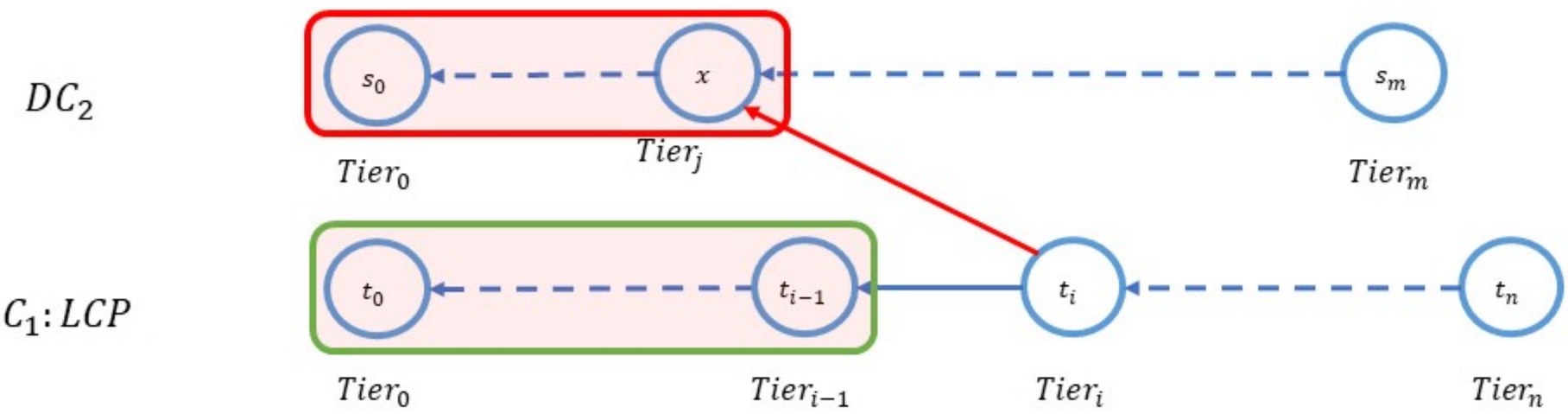
Outward Dependency

New dependency added $(t_i \rightarrow x)$ from a LCP node



Outward Dependency

New dependency added $(t_i \rightarrow x)$ from a LCP node



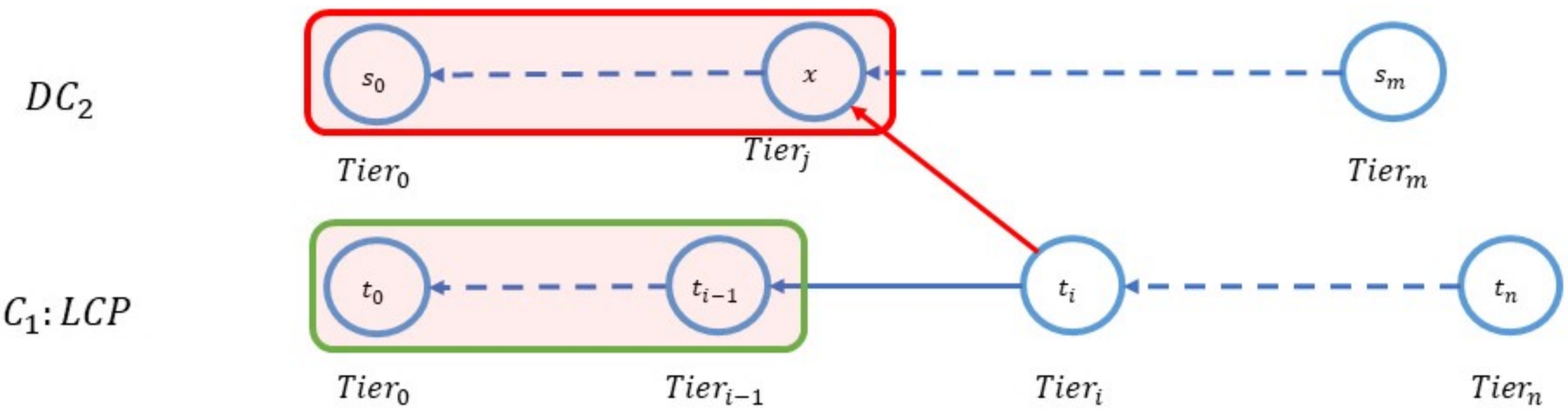
After the change:

IF $ExecTime(s_0, \dots, x) \leq ExecTime(t_0, \dots, t_{i-1})$

$LCP: DC_1$ (unchanged)

Outward Dependency

New dependency added $(t_i \rightarrow x)$ from a LCP node



After the change:

IF $ExecTime(s_0, \dots, x) \leq ExecTime(t_0, \dots, t_{i-1})$

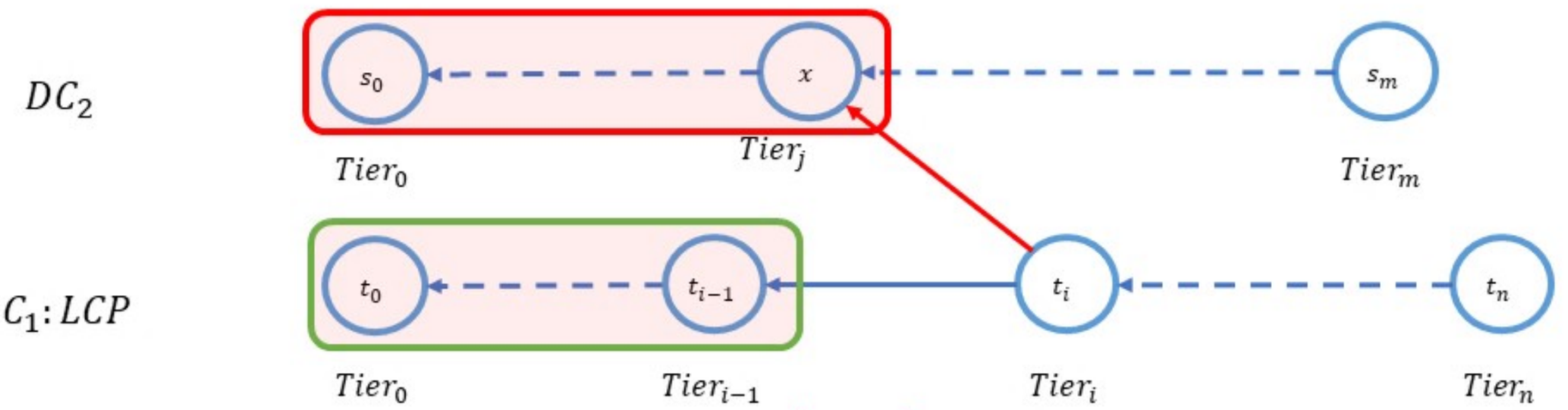
$LCP: DC_1$ (unchanged)

ELSE IF $ExecTime(s_0, \dots, x) > ExecTime(t_0, \dots, t_{i-1})$

$LCP: DC_3: (s_0, \dots, x, t_i, \dots, t_n)$

Outward Dependency

New dependency added $(t_i \rightarrow x)$ from a LCP node



After the change:

IF $ExecTime(s_0, \dots, x) \leq ExecTime(t_0, \dots, t_{i-1})$

$LCP: DC_1$ (unchanged)

ELSE IF $ExecTime(s_0, \dots, x) > ExecTime(t_0, \dots, t_{i-1})$

$LCP: DC_3: (s_0, \dots, x, t_i, \dots, t_n)$

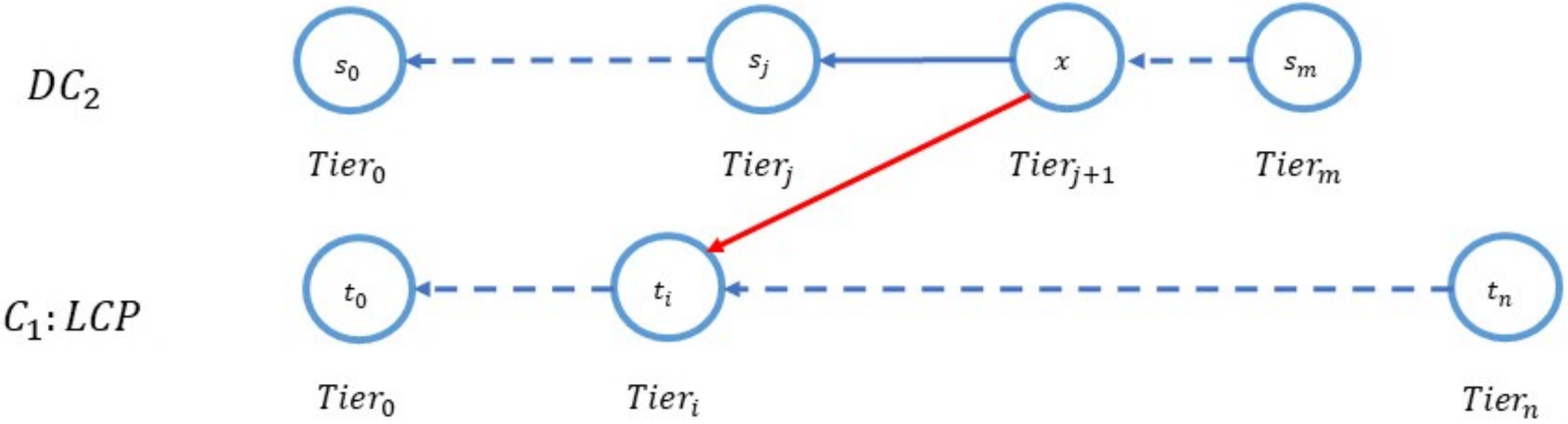
Change Impact:

Time increase: $ExecTime(s_0, \dots, x) - ExecTime(t_0, \dots, t_{i-1})$

Percentage of affected builds: $BuildCoverage(x)$

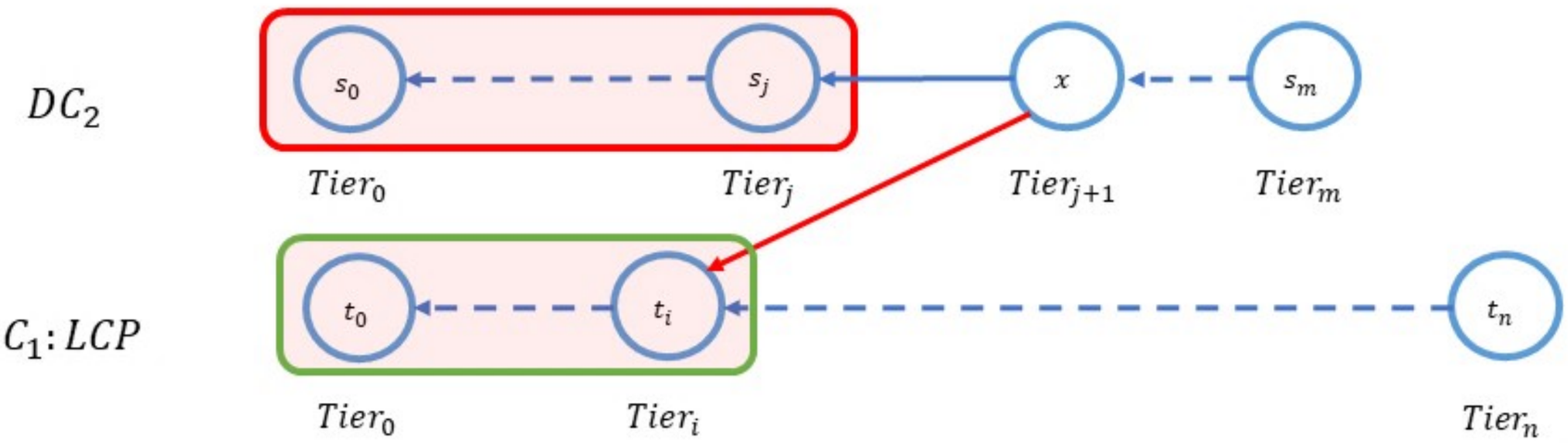
Inward Dependency

New dependency added $(x \rightarrow t_i)$ to a LCP node



Inward Dependency

New dependency added $(x \rightarrow t_i)$ to a LCP node



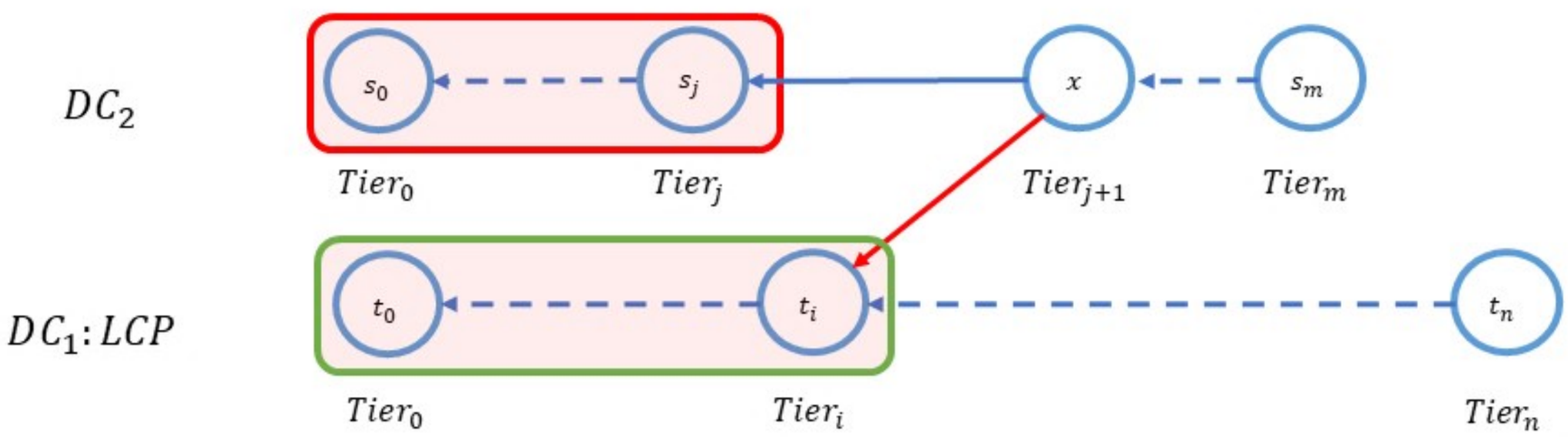
After the change:

IF $ExecTime(s_0, \dots, s_j) \geq ExecTime(t_0, \dots, t_i)$

$LCP: DC_1$ (unchanged)

Inward Dependency

New dependency added $(x \rightarrow t_i)$ to a LCP node



After the change:

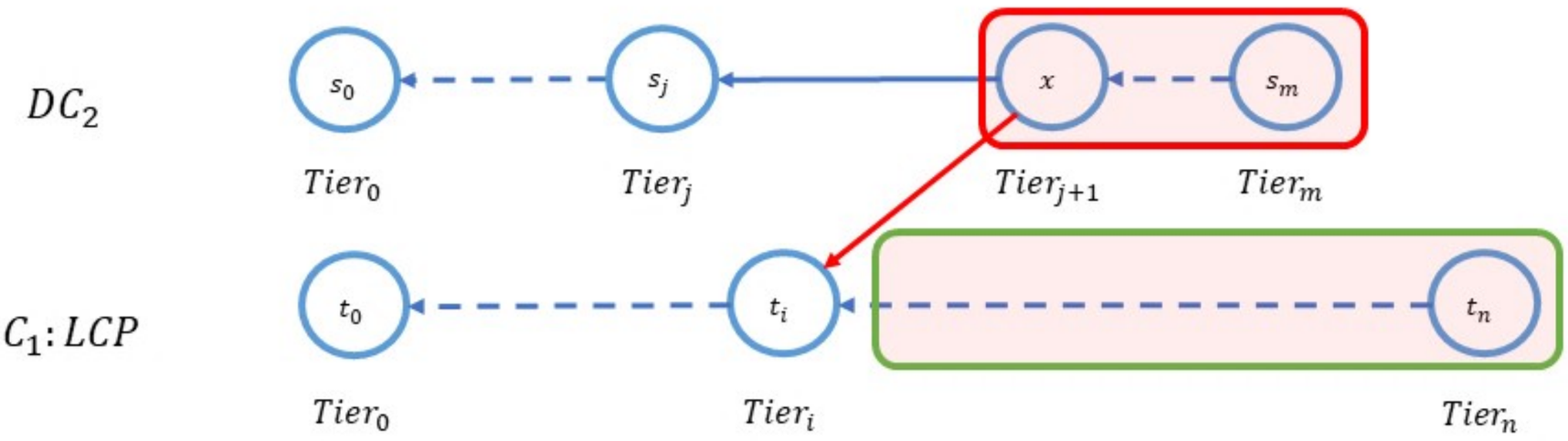
IF $ExecTime(s_0, \dots, s_j) \geq ExecTime(t_0, \dots, t_i)$

$LCP: DC_1$ (unchanged)

ELSE IF $ExecTime(s_0, \dots, s_j) < ExecTime(t_0, \dots, t_i)$

Inward Dependency

New dependency added $(x \rightarrow t_i)$ to a LCP node



After the change:

IF $ExecTime(s_0, \dots, s_j) \geq ExecTime(t_0, \dots, t_i)$

$LCP: DC_1$ (unchanged)

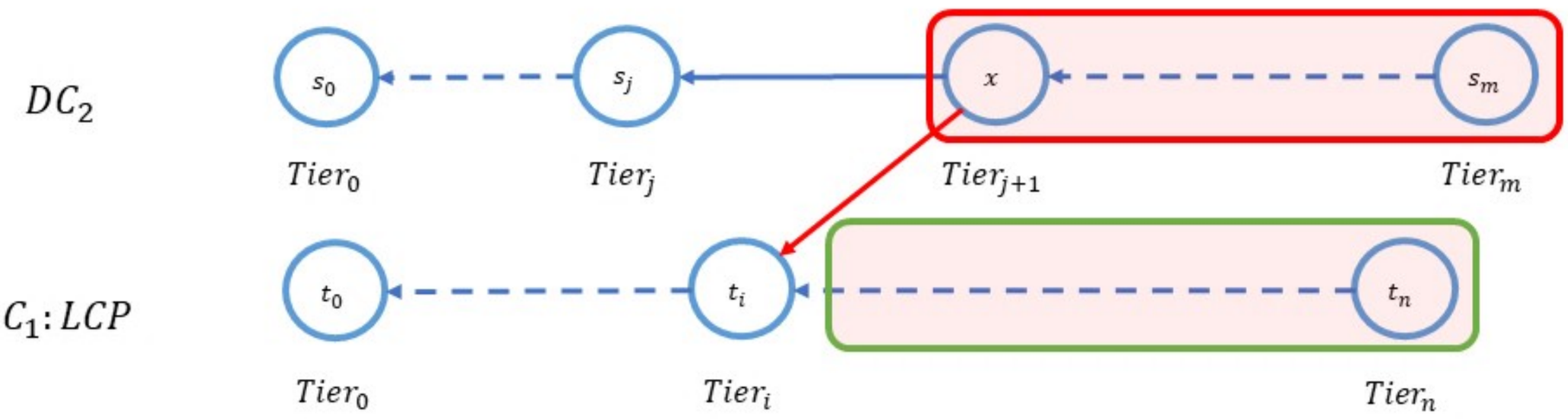
ELSE IF $ExecTime(s_0, \dots, s_j) < ExecTime(t_0, \dots, t_i)$

IF $ExecTime(x, \dots, s_m) \leq ExecTime(t_{i+1}, \dots, t_n)$

$LCP: DC_1$ (unchanged)

Inward Dependency

New dependency added $(x \rightarrow t_i)$ to a LCP node



After the change:

IF $ExecTime(s_0, \dots, s_j) \geq ExecTime(t_0, \dots, t_i)$

$LCP: DC_1$ (unchanged)

ELSE IF $ExecTime(s_0, \dots, s_j) < ExecTime(t_0, \dots, t_i)$

IF $ExecTime(x, \dots, s_m) \leq ExecTime(t_{i+1}, \dots, t_n)$

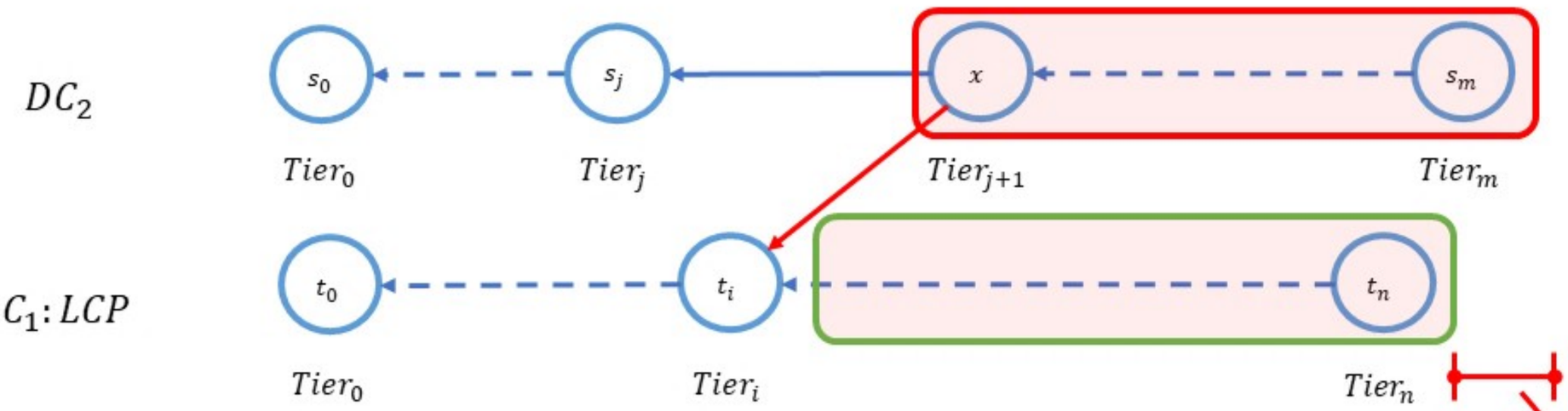
$LCP: DC_1$ (unchanged)

ELSE IF $ExecTime(x, \dots, s_m) > ExecTime(t_{i+1}, \dots, t_n)$

$LCP: DC_3: (t_0, \dots, t_i, x, \dots, s_m)$

Inward Dependency

New dependency added $(x \rightarrow t_i)$ to a LCP node



After the change:

IF $ExecTime(s_0, \dots, s_j) \geq ExecTime(t_0, \dots, t_i)$

$LCP: DC_1$ (unchanged)

ELSE IF $ExecTime(s_0, \dots, s_j) < ExecTime(t_0, \dots, t_i)$

IF $ExecTime(x, \dots, s_m) \leq ExecTime(t_{i+1}, \dots, t_n)$

$LCP: DC_1$ (unchanged)

ELSE IF $ExecTime(x, \dots, s_m) > ExecTime(t_{i+1}, \dots, t_n)$

$LCP: DC_3: (t_0, \dots, t_i, x, \dots, s_m)$

Change Impact:

Time increase: $ExecTime(x, \dots, s_m) - ExecTime(t_{i+1}, \dots, t_n)$

Percentage of affected builds: $BuildCoverage(t_i)$

Conclusions

Build Time Regression

- Threat for fast software delivery
- Difficult to diagnose and correct

Predict Build Impact

- Provide contextual info during the Pull Request process
- Allow early corrective operations

Future Work

Evaluation

- Run in shadow mode and estimate impact
- Evaluate prediction accuracy

Positive Feedback

- Positive impact on build activities
- Estimate reduction in build time

Questions?

Towards Predicting the Impact of Software Changes on Building Activities

M. Tufano, H. Sajnani, K. Herzig