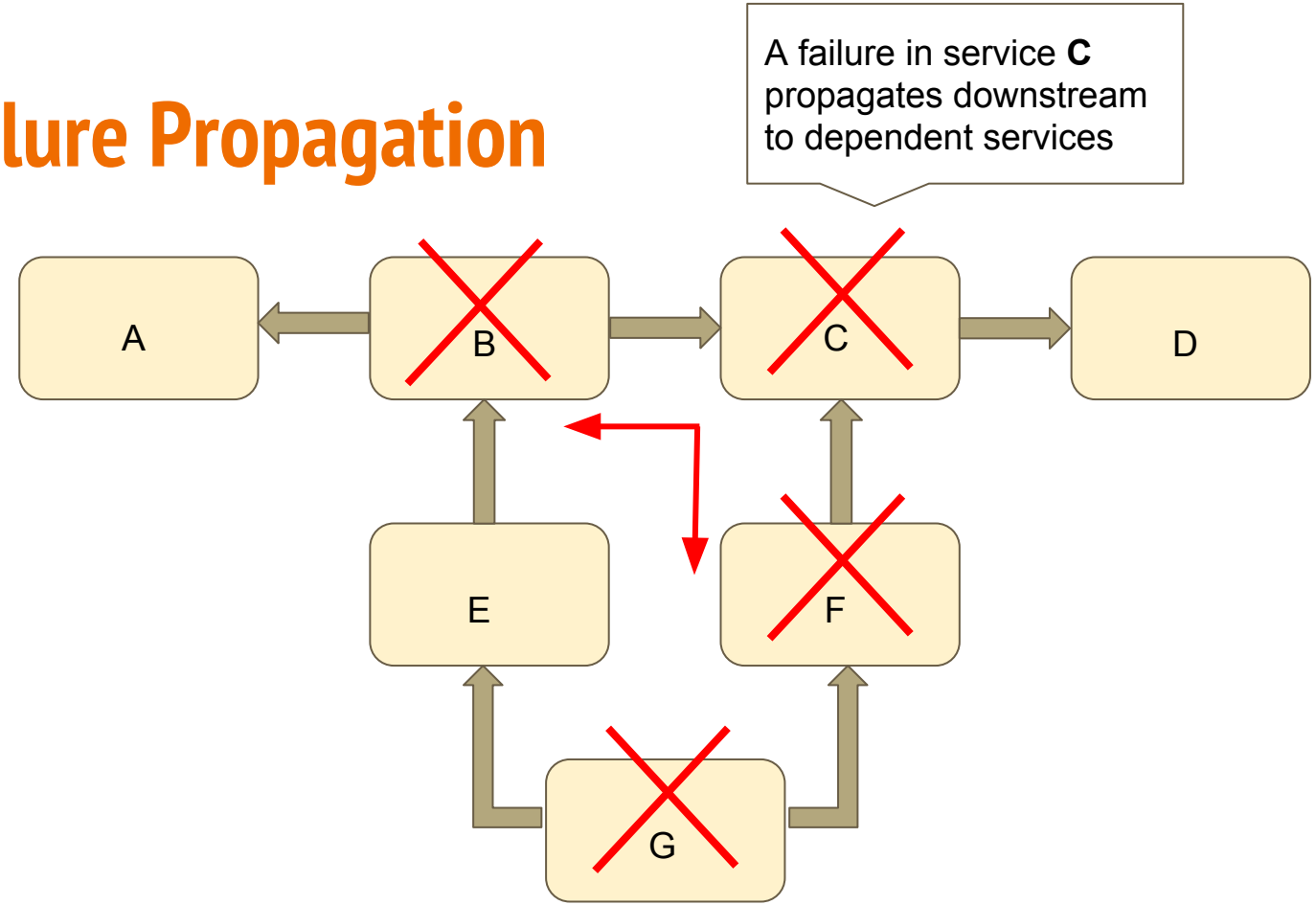
Managing APIs

CMPU4023 - Enterprise Application Development

Managing APIs

- The API publisher has the responsibility to ensure that the consumer has the best possible experience using the API
- The consumers of service APIs in the enterprise may be implementing business-critical functions, the failure of which could impact adversely on business continuity
- In an SOA environment, failures in upstream services may (and usually do) cause failures to propagate to downstream dependent services
- While by no means the only reason, one of the most common sources of errors occurs when changes to an API doesn't fully appreciate some of its behaviours being depended on by consumers - including bugs!

Failure Propagation



Versioning

- The idea behind versioning is to provide a shorthand to API consumers regarding which features of the API are supported and how they work
- This recognises that APIs do change and that consumers need to be aware of those changes and be able to make an assessment of impact
- Versioning schemes take many forms and which one is best depends on the circumstances and particulars of a service and its consumers
- A good scheme should convey the following information:

1. **API stability** - meaning how likely it is to change and be relied upon
2. **Major changes** - that new features have been added or existing ones changed
3. **Minor changes** - that existing features have been updated (e.g. bug fixes)
4. **Build identifier** - pinpoints the precise origins (contributing codebase) of the API version (often related to the underlying source code control system)

Version Compatibility

- The idea of API compatibility is to allow for the interoperability between one version of an API and a past version of itself or a future version of itself
- There are two kinds to consider:

1. **Backward compatibility** - changes to the API still allow legacy API consumers to transparently interoperate with the new version as if it was the old version (i.e. the client cannot distinguish between them). This type is usually critical

2. **Forward compatibility** - the API is designed in such a way that it will transparently interoperate with a future version of itself allowing clients using a new version of the API to work with legacy services at least to the extent of the functionality of the functionality offered by the legacy API

Changes and Stability

- Chief among the responsibilities of a good enterprise API is maintaining stability with consumers
- That usually means maintaining backward compatibility (BC) with all API changes affecting existing interfaces currently being relied upon by downstream clients - in which case the API is considered **stable**
- A change that does not maintain BC within the API is called a **breaking** change
- Breaking changes are often necessary but should be confined to early stage development or major version revisions
- Legacy version services often run concurrently with newly versioned service to help manage smooth service and API version transitions

Change Control

- We've noted already that enterprises tend to be conservative in the face of change
- Changes to APIs can knowingly or unknowingly break dependent consumers, the consequences of which can be very damaging
- Before any change is sanctioned, the project team must justify the proposed benefit to the business and have a well managed process for implementing, testing and backing out the change if necessary
- In the enterprise and particularly around business-critical functions, there is often a large multi-stakeholder team involved in risk assessment, benefit analysis and change process management

Summary

- API management and version control are important activities within service development and maintenance in the enterprise
- In a SOA environment, services which are often developed separately have interdependencies which are exposed through their respective APIs
- Service developers should maintain stable, backwardly compatible APIs to minimise the risk of unintentional interfaces breakage
- Where breaking changes do need to be made, these should be fully justified and managed accordingly