THE LONDON SCHOOL OF ECONOMICS AND POLITICAL SCIENCE

Technical

LSE Policy on Deciding to Write Software Code v1.1

The context and rationale for this policy is outlined in the Appendix.

Scope

1.1 In Scope

Software code written in DTS, business led technology teams within the School and any other areas who develop code are all in scope for this policy.

This document is complementary to the School's Software Development Policy. In terms of the software development process, this policy comes before the Software Development Policy as it covers deciding when to write code, rather than the details of how it should be written.

This document applies to:

- Code written for applications, whether those applications are business applications or are technical / operational applications that users do not directly engage with.
- Code written by LSE employees, contractors or third-party technology companies
- Bespoke code being re-written.
- Code written for integration purposes.
- Scripting

1.2 Out of Scope

The following areas are out of scope:

- Software code written for research purposes or as part of teaching activities.
- 'Declarative' programming. This can be defined as statements which define what the coding needs to achieve, rather than directly specifiying how it should achieve it. It is not subject to this policy. While it still costs time to write, crucially, it can be upgraded automatically, so it is not subject to the same ongoing costs of maintenance.
- Code written or generated to deploy infrastructure in the cloud.
- This policy does not apply to code that was completed before this policy was approved (28th February, 2020). However, future changes to code already in use is in scope.

Approval to Code

This document is based on three key principles:

1. Coding is a last resort
2. We consider the full lifetime cost of writing software.
3. We do an active review of the decision to write software.

To do this, there is a three-tiered approach. It is based on the amount of development effort required. In this context, that includes time required in analysing, coding, testing and deploying the code.

1. Development effort is between 1 and 5 days

The developer, or their manager notifies the DTS Head of Design and Build of the work being undertaken. This includes:

- The software to be written.
- A brief description of the changes to be made (eg 100 – 200 words).
- An outline of the alternatives considered.

The DTS Head of Design and Build keeps a record of all notifications, for review by the Developers Community of Practice. S/he also monitors to make sure that larger developments are not being broken down into small components. This record is reviewed at least once a quarter by the Head of Strategy and Architecture.

2. Development effort is from 6 days to 20 days

The developer, or their manager, brings the proposed change to the Developers' Community of Practice for discussion. They notify the group beforehand, including the details above, plus

- A justification of why none of the alternatives considered were acceptable.
- An explanation of how the support, maintenance, patching, hosting and decommissioning for the coding change will be funded over its lifetime.

3. Development effort is more than 20 days

The manager brings the proposed change to the Architecture Board for review. They provide the Architecture Board with the information above plus:

- A business analyst from the BIU reviews all requests to code that are more than 20 days coding effort, to make sure that there is genuinely no realistic alternative. This is a brief (e.g. 1 day) investigation.
- A comparison of the lifetime costs of the software development, contrasted with any extra business process costs involved. Note that:
    - This includes analysing, developing, testing, deploying, hosting, supporting, maintaining and decommissioning that code.
    - It includes all of the costs and effort needed to maintain the whole technical stack used to deploy the code, for example including middleware, database, file storage.
    - The minimum lifespan of the code should be estimated as seven years.
    - Where code is written or supported externally, the vendor's daily rate or an average £600 per day is used, whichever is higher.

The Architecture Board considers the request to develop. It may choose to ask the manager to come to the meeting to discuss the proposed change. If a decision needs to be made before the next scheduled meeting, the approval may be done offline.

4. Breaches of the Process

Breaches of this process are managed as Architecture Exceptions. The process for managing exceptions is described in the document “Architecture Exceptions”, available from the DTS Strategy & Architecture team. The process was approved by the School’s Architecture Board on 4th June, 2020.

In brief, all potential architecture exceptions will be notified to the Assistant Director for Strategy and Architecture in DTS who will give advice about next steps. This is likely to include notification to the School’s Architecture Board and, in certain circumstances, to the School’s Data, Technology and Digital Management Board.

Appendix

1. Context

We have a considerable, and expensive legacy of out of support and poorly understood code. This has led us to an increasing number of problems supporting, changing and decommissioning bespoke software.

The technology sector has moved away from coding as the first option in areas where there is little or no competitive advantage. This is addressed in the School's Architecture Principles, especially in Principle 9: Reuse over Rent over Buy over Build. This principle states that:

> "When the School has a new requirement, our first course of action is to use functionality from an existing solution. If no existing solution can meet the requirements, we will 'rent' software (ie use 'software as a service'). If there is no suitable SaaS solution, we will buy and integrate commercial off the shelf solutions or package components. If there is no viable solution on the market we will develop a bespoke solution, but only if the risk profile, scale and total cost of ownership are explicitly documented and deemed acceptable by the Architecture Board and the Portfolio Board."

2. The Cost of Coding

The full cost of writing software code is often underestimated. People think that something that is estimated as five days to code will take five days effort and then be implemented for ongoing use. However, that is a major underestimation for two reasons:

- A lack of understanding of the activities required.
- The fact that, within software development, effort estimates are often over-optimistic and there is frequently over confidence in their accuracy.

2.1 Implementation Effort

The actual time taken to write code for a new piece of functionality is only a portion of the full effort needed to get the software live. Typical development activities, in order, are:

Requirements → Design → Code → Test → Deploy & Transition

(While this will be more iterative for an agile development, the basis of it holds true.) Coding is just one of the main activities.

It is a reasonable 'rule of thumb' to regard coding as around 25% of the effort needed to get something live. This suggests that something that "Only takes five days to code" will need around 20 days effort to implement fully.

Additionally, when we write code, we carry the risk of that effort not being able to deliver the code as expected or of it needing additional effort later. It is generally more risky to code than to meet that requirement in any other way.

2.2 Lifetime Cost of Ownership

- **Cost to support and maintain:** Based on existing LSE code, assume that any code written within the School will have an average lifetime of a minimum of seven years. In some cases it will be considerably more. During this lifetime the software will need to be maintained to keep it up-to-date with new releases of supporting databases, coding frameworks, OSs, etc. It will also need to be 'bug fixed' to address problems. Security holes will need to be patched. Additionally, it may need to be enhanced and extended.

A very rough rule of thumb is that the cost to implement a piece of software is 20% of the full lifetime cost of that software. This suggests that the small new piece of functionality that originally took only 5 days to code, will take around 20 days to implement and around 100 days of support and maintenance effort over its lifetime.

In addition, the more you change a piece of software code over time, the more it costs to make changes, for example because complexity is added, different programmers are often involved, techniques are more likely to vary.

- **Hosting cost:** when we write software we have to host it. In a few cases we may be able to make use of an existing service or platform, but normally there will be additional costs to us, for hardware, hardware support and maintenance, operating system and database licenses, database administration, disaster recovery etc.

- **Decommissioning**: The LSE often does not consider decommissioning costs and they are left out of project costs. The School's guidelines on decommissioning now state that "Projects will consider decommissioning as part of the requirements gathering stage of the project."

Taken together, the implementation effort and the lifetime cost of ownership underline that the number of days' coding effort is a very poor way of estimating the total effort needed to develop and own new functionality.

2.3 Product Ownership

Within the LSE we have often had a piecemeal approach to software development in that someone asks for a new piece of functionality, so we look into it and deliver it. We do not normally have a product strategy or clear product ownership and consider how the proposed change sits within the product plans and strategies. This means that we are more likely to make short term decisions which may then hamper or interfere with strategic objectives. We also do not usually apply budgets by products, meaning that getting money can happen in a first come / first served way.

2.4 Fit with Requirements

One of the major benefits of bespoke code is that it can have a very strong fit with requirements. However, there are sometimes unrealistic expectations within the LSE of what is an acceptable level of fit with requirements. One project stated in their business case that they must have a "100% fit with requirements". This is an unrealistic and impractical expectation; externally a 70 – 80% fit with requirements would be considered a very good match.

Review schedule

Review interval	Next review due by	Next review start
v1.0a	28/2/21	15/2/21
v1.1	12/6/2020	1/6/2020

Version history

Version	Date	Approved by	Notes
v1.0a	28/2/2020	Architecture Board	
v.1.1	12/6/2020	Architecture Board	

Links

Reference	Link
-	-
-	-

Contacts

Position	Name	Email	Notes
Asst Dir for Strategy & Architecture	Michael D'Urso	m.g.durso@lse.a.cuk	

Communications and Training

Will this document be publicised through Internal Communications?	Yes/ **No**
Will training needs arise from this policy	Yes/ **No**
If Yes, please give details	