Assessment of Software for Government

Version 1.0, April 2012

Aim

1. This document presents an assessment model for selecting software, including open source software, for use across Government, and the wider UK public sector.

2. It is presented in recognition that potentially better value for money software, including open source software, is underused across Government and the wider public sector. It aims to address the lack of a consistent approach to the selection of software across Government, and to improve the transparency of these selection processes undertaken by Government or suppliers to Government.

3. In particular, it aims to emphasise that open source software is considered against the same selection criteria as closed proprietary software, whilst also explaining how these same criteria can be evaluated in the context of the different open source development model and ecosystem.

4. This document does not aim to encourage only the selection of software which shows topmost indicators for each criterion, but instead the more appropriate matching of software characteristics with actual risk and requirements. It is important that risk and requirements are justifiable, as these can drive significant unnecessary cost.

5. This document also covers some characteristics of service providers and integrators, particularly with respect to open source software. The aim is help select service providers and integrators by making clear the capabilities and behaviours which can enable Government to derive value from open source software.

Context

1. The Coalition Government believes open source software can potentially deliver significant short and long term cost savings across Government IT. This document is part of the Open Source Toolkit developed as an action of the UK Government ICT Strategy 2011.

2. Typical benefits of open source software include lower procurement prices, no license costs, interoperability, easier integration and customisation, compliance with open technology and data standards enabling autonomy over your own information and freedom from vendor lock in.

3. Open source software is not currently widely used in Government IT, and the leading systems integrators for Government Departments do not routinely consider open source software for IT solution options, as required by existing HMG ICT policy.

4. There are significant and wide ranging obstacles to open source software in Government. Some of these are a lack of clear procurement guidance, resistance from suppliers, concerns about license obligations and patent issues, misunderstanding of the security accreditation process, and myths around open source quality, its development and support ecosystem.

How To Use

1. This document presents criteria for assessing the suitability of software for a particular business requirement. It extends previous work by explaining how these criteria can be evaluated in the context of open source software, its development model and ecosystem. It also clarifies criteria which should not be used to assess the suitability of software.

2. The primary audiences for this assessment model are technical and enterprise **architects, commercial / procurement officers** and **project managers** within the civil service, and those from the **supplier** and **integrator** community who influence the design and makeup of ICT solutions to Government. Customers and suppliers in the wider public sector are also encouraged to make use of this document.

3. This set of assessment criteria can be used to:
   a. **Inform** the design of new IT solutions.
   b. **Suggest** opportunities for IT service or solution refreshes.
   c. **Challenge** a proposed solution that does not use open source technology.

4. Software under consideration does not need to conform to the topmost positive indicators listed for all criteria. This assessment model will help Government challenge themselves, and suppliers, that potentially better value for money software has sufficient positive characteristics to meet justifiable business requirements and risk position. It is not expected to be used to argue for more expensive IT solutions. For each criterion, the reader is encouraged to ask "**What level of indicator is really required? Is the requirement justifiable?**"

5. This document does not remove existing requirements for due diligence and assurance on the part of Government. In particular it does not transfer any technology risk from IT integrators and suppliers to Government, where it has previously been contractually placed with those suppliers and integrators. It also does not supersede any government standards for security and accreditation.

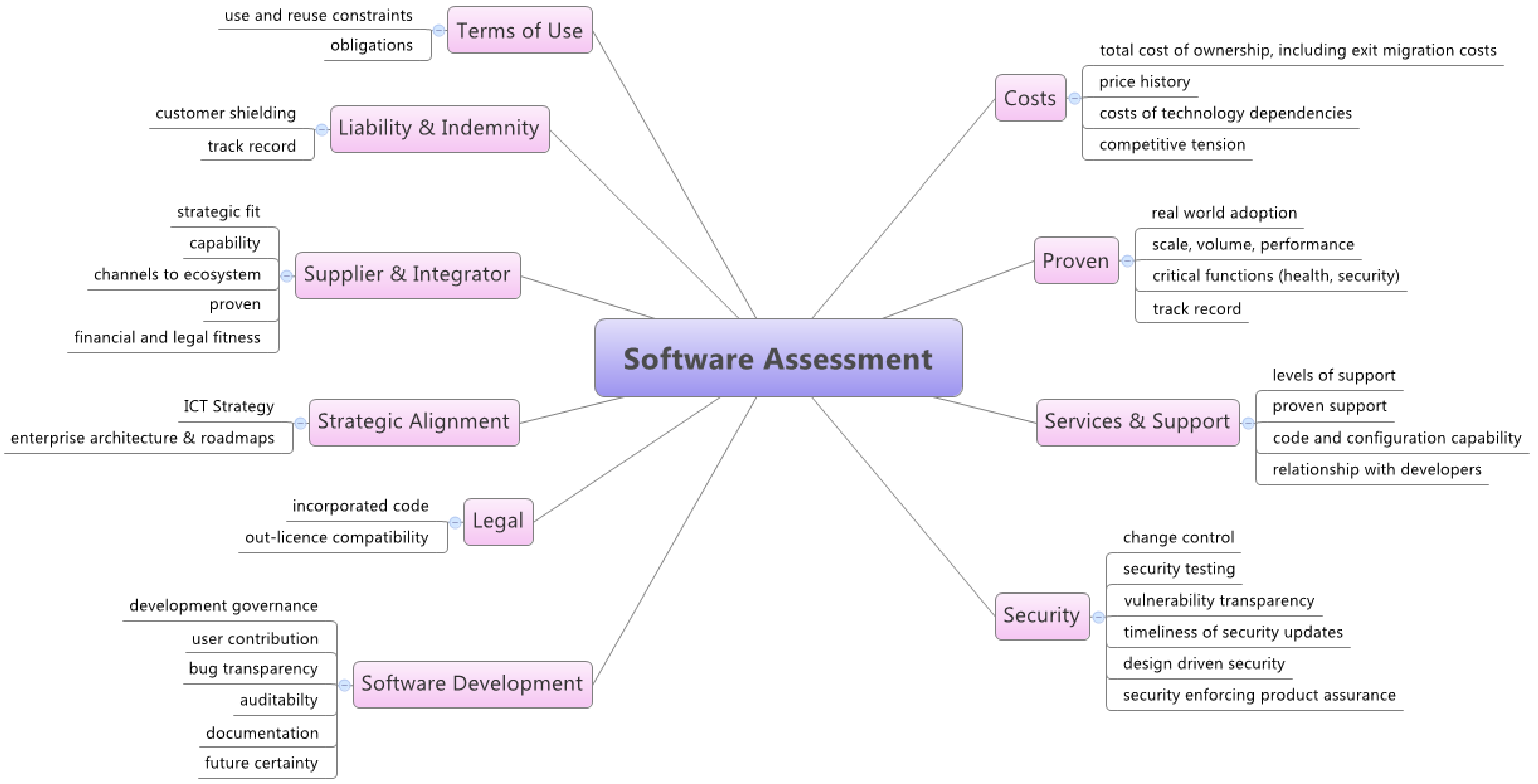
6. This document should be used in conjunction with the wider Open Source Toolkit published via the Cabinet Office website. Government, public sector, existing and potentially new suppliers are encouraged to engage the open source surgery process to discuss queries or issues, via opensource@homeoffice.gsi.gov.uk.

7. This model for software selection has been developed in consultation with the SI Forum, a cross government board established as an action of the UK Government ICT Strategy 2011. Its members include **Atos, BT, Fujitsu, HP, IBM, Logica, Serco, Sirius** and **Steria**. It is not expected that this assessment model conflicts with, or deviates significantly from, their internal assessment processes.

Feedback & Suggestions

Please provide feedback and suggestions to opensource@homeoffice.gsi.gov.uk

Overview of Software Selection



Principles for Software Selection

The following key principles are highlighted to address current known barriers to the use of better value software, including open source software.

- **Value & Cost** – the value and cost of software is realised beyond its initial deployment, particularly through dependencies on other systems. Refer to the London School of Economics report and the guide to software total cost of ownership, both published as part of the Open Source Toolkit. In particular, the practice of comparing the only purchase cost of software is deficient, and risks being subject to artificial manipulation of up front pricing. It also omits potentially critical in-life change, and end of life exit and migration costs.

- **Risk** - Software and the services supporting it should be selected to meet actual and justifiable business risk, and no more. Overestimating risk can significantly drive up unnecessary costs. Risk should be unbundled to better match different kinds of software sourcing to optimise for value.

- **Requirements Challenge and Commodity** – Business requirements should be challenged if their adjustment can lead to better value from different software solutions. Furthermore, the greater the divergence of business requirements from the wider market, the more bespoke IT solutions become, which drives cost. Commodity IT means commodity requirements, commodity technology, and also the commodity use if that technology.

- **Technical Refresh** – It is expected that software options are evaluated not only for new builds, but also at technology refresh points. Technology refresh triggers should not only be used to upgrade existing software to more current versions.

- **Reuse and Diversity** – It is a common strategy of government departments to reuse software and reduce technology duplication and diversity. However this strategy must be applied to optimise value. This means not reusing or extending technology which is not aligned to the ICT Strategy or a desired future architectural state. It also means that multiple software for similar functions may be more optimal if sufficient use cases do not require the more expensive software. For example, it can be better value for a licence-free open source application server used for almost all such departmental functions, with an expensively licensed application server for the very few exceptional requirements.

- **Security** – Open source software is no more or less secure, as a category of software, than closed proprietary software. CESG GPG 38 discusses this and related issues around the evaluation of risk for software including open source software. This means open source software cannot be excluded from an options analysis on the premise that it is less secure than closed proprietary software. It is a misunderstanding of the accreditation process to assert that some software is pre-accredited. For further discussion see CESG GPG 38 and the associated security note published with the Toolkit.

- **Brand Recognition & Innovation** – Brand recognition on its own must not be used to justify the procurement of any software.

Assessment of Software for Government

Criterion	High	Intermediate	Low
Proven	• There are many real world examples of the software in use. • There are existing uses of the software in Government.	• There are some real world examples of the software in use. • The software, though in use, has not been used in Government or the wider public sector.	• There are no real world examples of the software in use. • There are only insignificant uses of the software.
	• The software has proven significant use with respect to some or all of: ○ performance ○ large scale ○ critical function (eg safety, security)	• The software has proven moderate use with respect to performance, scale or criticality.	• The software has not been demonstrated in the real world under any significant use with respect to performance, scale or critical function.
	• The software has been successfully used over a long term with respect to ICT product cycles.	• The software has been successfully used over a medium term with respect to ICT product cycles.	• The software is new and has not been proven over any significant time to ascertain its operating characteristics.
	• Publicly available performance and scaling benchmarks, including repeatable test configuration are available, with results publicly shareable for analysis.	• Software supplier publishes performance and scaling indicators, but these are not publicly repeatable, due to insufficient test configuration detail, for example.	• No performance or scaling data available.
Services & Support	• Support exists at several levels, including for mission critical services if needed.	• Software is supported but not for mission critical services. For open source, community support can be sufficient.	• No support arrangement.

	• Support has been proven over a number of years. • Support has been proven over a range of sectors, including demanding sectors such as finance and Government.	• Support is new and has not been proven over a number of years. • Support has only been proven over a limited range of sectors.	• No support.
	• Software support includes bug fix and code change capability, as well as configuration support.	• Support capability only covers configuration support.	• No support.
	• Support supplier is, or has productive links to, the software developers, or development community.	• Support supplier is not the software developer, nor has productive links to those developers, or development community.	• No support.
Security	• Changes to software are strictly assessed, audited and controlled through mature governance. This applies to commercial software organisations as well as community development. • Third party assurance covers change control and governance, not just testing the resultant software.	• Changes to the software are limited to selected developers who have proven their ability to maintain software quality. These trusted developers have write / commit access to the code repository. • Ad hoc contributions to the software are channelled through these selected developers for assessment and approval.	• Changes to the software are subject to minimal, or no, assessment and governance.
	• Software is subject to regular security and penetration testing undertaken by the developer. Outcomes drive security fixes. • Software is security / penetration tested and assured by an independent and trusted third party.	• Software is regularly security tested by third parties.	• No software security testing.
	• The software supplier is fully open and	• The software supplier selectively	• The supplier does not discuss

	transparent about publicising known vulnerabilities and exploits. Impact assessment of vulnerabilities and exploits is issued. • All known vulnerabilities are acknowledged by the supplier.	publicises vulnerabilities and exploits. • The supplier does not acknowledge all known vulnerabilities.	vulnerabilities or exploits. • The supplier does not acknowledge vulnerabilities discovered by the wider community.
	• The time to fix vulnerabilities is minimal and timely, minimising exposure to exploits. • All known vulnerabilities are fixed.	• The time to fix is moderately timely. • Not all known vulnerabilities are fixed.	• Time to fix is long, or no fix is issued. • No management of vulnerabilities and fixes.
	• Design and architecture decisions taken to support software security. For example, use of minimal privilege, privilege isolation, and defence in depth across development, and approaches such as static code analysis.	• Normal software design and development practices with no design or architectural decisions to support software security.	• No software design and development discipline.
	• For security enforcing products, the software has received an appropriate certification of assurance, for example from CESG. This only applies to a limited set of software functions such as firewalls, virtualisation, VPN and encryption software.	• n/a	• No assurance of security enforcing function.
Software Development	• Software development is well managed and governed. • Software development is open to user contributions or feature suggestions, but these are assessed and only enter the code in a well managed and governed manner.	• Software development is loosely managed, with change control but minimal coherent direction or quality criteria. • Software development is arbitrarily open to contributions from the user community but there is no formal assessment or governance around this.	• Software development is undisciplined, not following any process or governance. • Software development is not open to public contribution.

	• Software issues and bugs are managed by a bug tracking and resolution system that is transparent and has public visibility.	• Software issues and bugs are managed internally, with no public visibility.	• There is no coherent approach to tracking issues or bugs to resolution.
	• Software development is fully auditable for code changes, their source, and the reason for change.	• Software development is change controlled but with limited auditability for code changes, their source, and the reason for change.	• Software development is not auditable.
	• Software is fully documented and documentation is updated and available when required.	• Software is documented but updates sometimes lag behind software development. Documentation is available when required. • Alternatively, software is sufficiently well written and commented to allow non-specialist developers to understand its workings.	• Software is not documented or is documented in an ad-hoc manner. • Software is not sufficiently well coded and commented to allow non-specialist developers to understand its workings.
	• Software development broadly follows published roadmaps. It is recognised that roadmaps evolve.	• Software development, though well managed, does not intend to provide predictability into the medium to long term. Only short term or immediate changes are predictable.	• No predictable development direction or roadmaps.
Legal	• Software is developed with full understanding and tracking of incorporated code from third parties, patents and inherited license obligations. • Documented clarity around compatibility of licenses for constituent software (in-licences) and compatibility with licence of resultant product (out-licence).	• Software is developed and documented in a fairly well controlled manner to provide confidence, but not guarantees, with respect to incorporated code from third parties, patents and inherited license obligations.	• Software is developed with no understanding of incorporated code from third parties, patents and inherited license obligations.

Liability & Indemnity	• Customer is shielded, or indemnified, from legal action related to patents, licenses or other issues by the software support vendor or the integrator. This can be done directly from the software developer, or indirectly through third party insurance. • Software has a track record of no significant legal challenges. • The open source software has successfully defended against legal action, with respect to copyright, patents or other issues, in a comparable jurisdiction, thus setting a legal precedent.	• Customer has to undertake its own due diligence for legal liability around open source software, because it cannot be obtained from the software support vendor or the integrator. In the event of legal action, it must defend itself. • The software is subject to ongoing but as yet unresolved legal action, or is the subject of negative commentary from commercial rivals. Note that this can be common in the software sector, with legal action and commentary sometimes used for competitive advantage and to influence customer behaviour.	• Customer cannot obtain any assurances around legal liability for the open source software in question. • The open source software has failed in defending against legal action, with respect to copyright, patents or other issues, in a comparable jurisdiction.
Terms of Use, License or Contract	• Software terms allow use for primary requirement, and also for varying requirements or use cases. • Software terms allow subsequent reuse across Government, including for varying requirements. • Software terms allow use across any sector, for any requirement, as per many open source licenses. Note open source licenses are generally terms of use, not units of sale for payment.	• Software terms only allow use for specific requirement. Use for varying requirements is not permitted. • Software terms do not allow reuse across Government.	• Software terms forbid use for, or are incompatible with, primary requirement.
	• Software terms places no onerous obligations on customers, subsequent to its use. Most open source licenses, including the GPL, MIT, Apache and BSD licenses are not onerous for expected Government use cases.	• Software license places some obligations or constraints on customers, subsequent to its use. Licenses which require the public release of sensitive code fall into this category. Most open source licenses do not require such release for expected Government use cases.	• Software license places onerous obligations on customers, subsequent to its use. This is rarely true for common open source software. • For example, the rare Affero GPL variant requires making available the full source code for an application which is used over

			a network, to those users. This is distinct from the more common GPL where only redistribution of the executable beyond an organisation's boundary triggers the need to share the source code.
Costs	• Software does not impose costs by creating or enforcing dependencies to other software or technologies.	• Software imposes costs by creating or enforcing dependencies to other software or technologies. Extent of dependencies into infrastructure is moderate or limited.	• Software imposes costs by creating or enforcing dependencies to other software or technologies. Extent of dependencies into infrastructure is significant.
	• Exit costs from software are known and minimal. In particular, business information (content, metadata, workflows, for example) can be fully exported to a neutral format ready for migration to a subsequent technology, with ease and at minimal cost.	• Exit costs from software are known but significant. Business information can be fully exported to a neutral format ready for migration to a subsequent technology, but this process is expensive and requires specialist and rare knowledge or skills, and therefore a risk in itself.	• Exit costs from software are able to be determined. There is no identifiable mechanism to export business information to a neutral format such that it can be migrated to a subsequent technology.
	• Software's function is subject to realistic competition, ensuring software vendor is subject to competitive tension.	• Software's function is subject to minimal competition. Software vendor is subject only to mild competitive tension.	• Software's function is not subject to realistic competition, and software has effective monopoly, not subject to competitive tension.
	• Software has a price history which suggests no unexpected future price rises.	• Software has a history of periodic price rises, which cannot be budgeted for presently, but which will have to be accommodated as and when they arise.	• n/a
Strategic Alignment	• Software is aligned to the Government ICT Strategy and vision. This covers domains including commodity IT, competitive markets, accessibility to SME suppliers, open standards and interoperability, and value for money.	• n/a	• Software is in conflict with the Government ICT Strategy and vision.

	• Software is aligned to departmental enterprise architecture and technology roadmaps.	• n/a	• Software is in conflict with departmental enterprise architecture and technology roadmaps.

Suppliers and Integrators of Open Source Software for Government

Criterion	High	Intermediate	Low
IT Integrator or Supplier	• Integrator understands the software development model, and its ecosystem, including for open source.	• Integrator only understands proprietary software ecosystem. Integrator understands open source software and its ecosystem only enough to make occasional use of software, primarily imitating other integrators.	• Integrator does not understand open source software and its ecosystem, and has no intention to use it strategically.
	• Integrator's legal and commercial units understand open source software and are fully engaged with it "business as usual".	• Integrator's legal and commercial units partially understand open source software and only engage with it tactically and by exception.	• Integrator's legal and commercial units do not understand or engage with open source software.
	• Open source software plays a primary tier role in the integrator's strategic vision and approach to IT solutions and services. • Integrator maintains permanent internal expertise for strategic open source software.	• Open source software does not play a primary tier role in the integrator's strategic vision and approach to IT solutions and services. It only plays a secondary role where it is used tactically or by exception. • Integrator's internal expertise is incidental. Integrator buys in open source expertise on a temporary and case by case basis.	• Open source plays no part in the integrator's strategic vision or approach to IT solutions. • Integrator has no internal expertise in open source software, and does not engage temporary expertise.
	• Integrator has established channels to support and maintenance for open source software. • Integrator also has partnerships with design and integration specialists for specific open source software.	• Integrator engages channels to support and maintenance for open source software on a case by case basis, and often by exception. • Integrator only forms partnerships with design and integration specialists for specific open source software by	• Integrator has no channels to open source support and maintenance. • Integrator has no partnerships with open source software specialists.

		exception.	
	• Integrator has proven successful experience of open source software, over a wide range of solutions and services, including mission critical.	• Integrator has some proven successful experience of open source software, over a limited set of solution and service types.	• Integrator has no proven experience of open source software as part of its solutions and services.
	• Integrator is capable of managing open source software which has been modified to better meet the customer’s requirements. The modification is undertaken internally or through partnerships with 3rd party specialists.	• Integrator does not manage modified open source software, only using well known releases from upstream support suppliers and developers.	• n/a
	• Software developer, supplier and integrator are financially robust to the extent appropriate to the use of the software and associated business risk.	• Only the integrator is financially robust to the extent appropriate to the use of the software and associated business risk. The integrator will, in effect, cover for financial instability.	• Software developer, supplier and integrator are not financially robust to the extent appropriate to the use of the software and associated services.