EBMeDS Quality Plan

Process description

Last saved on 16 February 2012 by Rasmus Paetau
(Select above text and press F9 to update date and modifier fields!)

Table of Contents

Introduction

The **aim** of the EBMeDS Quality Plan is to ensure that the quality of the EBMeDS system is as high as possible. The concept of quality for this purpose has been defined in a separate document. The purpose of the document at hand is to identify the major quality influencing factors in our development process, and to suggest measures for maximising quality based on them. Influencing quality is approached mainly from the perspective of an individual script's production process, but important quality factors falling outside this process are also presented when they can be clearly identified.

The document is structured to first present the quality-affecting processes – divided into the production process of an individual script on the one hand, and processes of a more general nature on the other – along with general observations on their importance with respect to quality. This general presentation of processes is followed by concrete quality oriented measures, listed in table format, that are matched to the previously presented processes. A glossary, clarifying certain concepts that have been printed in *italics*, can be found at the end of the document.

Processes affecting quality

The following sections will consider the processes involved in the development and maintenance of the EBMeDS system from a quality perspective. Processes related to the development and maintenance of individual scripts are examined first, while more general processes concerning the entire script collection and the EBMeDS system are presented later.

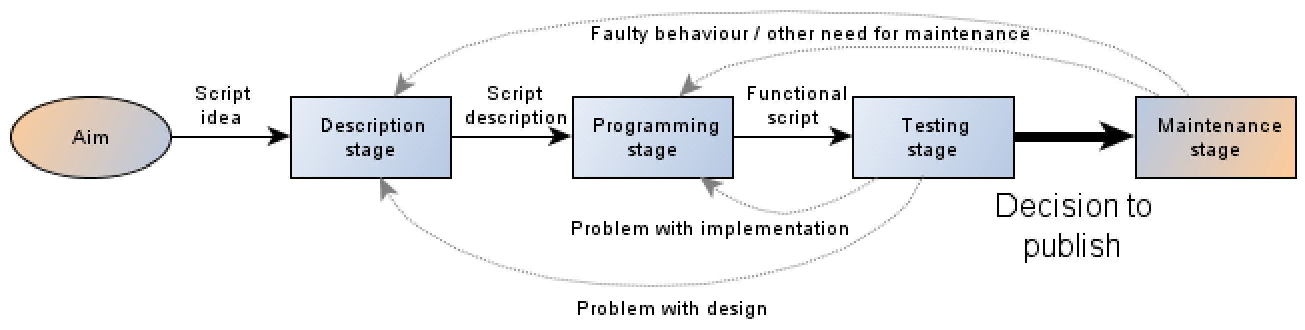
The script production process

The script production process starts with a *script idea*, which has usually arisen as a response to a specific problem or need. The aim may be to

- support clinical decision making by serving up evidence- or experience-based information and conventions such as evidence summaries, clinical practice guidelines, and locally agreed-upon care pathways,
- accelerate repetitive tasks and clinical information processing e.g. by collecting data for the purposes of writing medical certificates, computing risk scores relevant to current decision-making, or producing informative at-a-glance views of relevant data
- draw attention to obvious errors and hazards, and thereby reduce mistakes arising from deficiencies in either knowledge or process design

The production process onwards from the script idea can be divided into four main stages, which we will refer to as the stages of description, programming, testing and maintenance. Figure 1 below presents a crude outline of how these stages relate to each other and the script production process.

Figure 1: Stages of the script production process



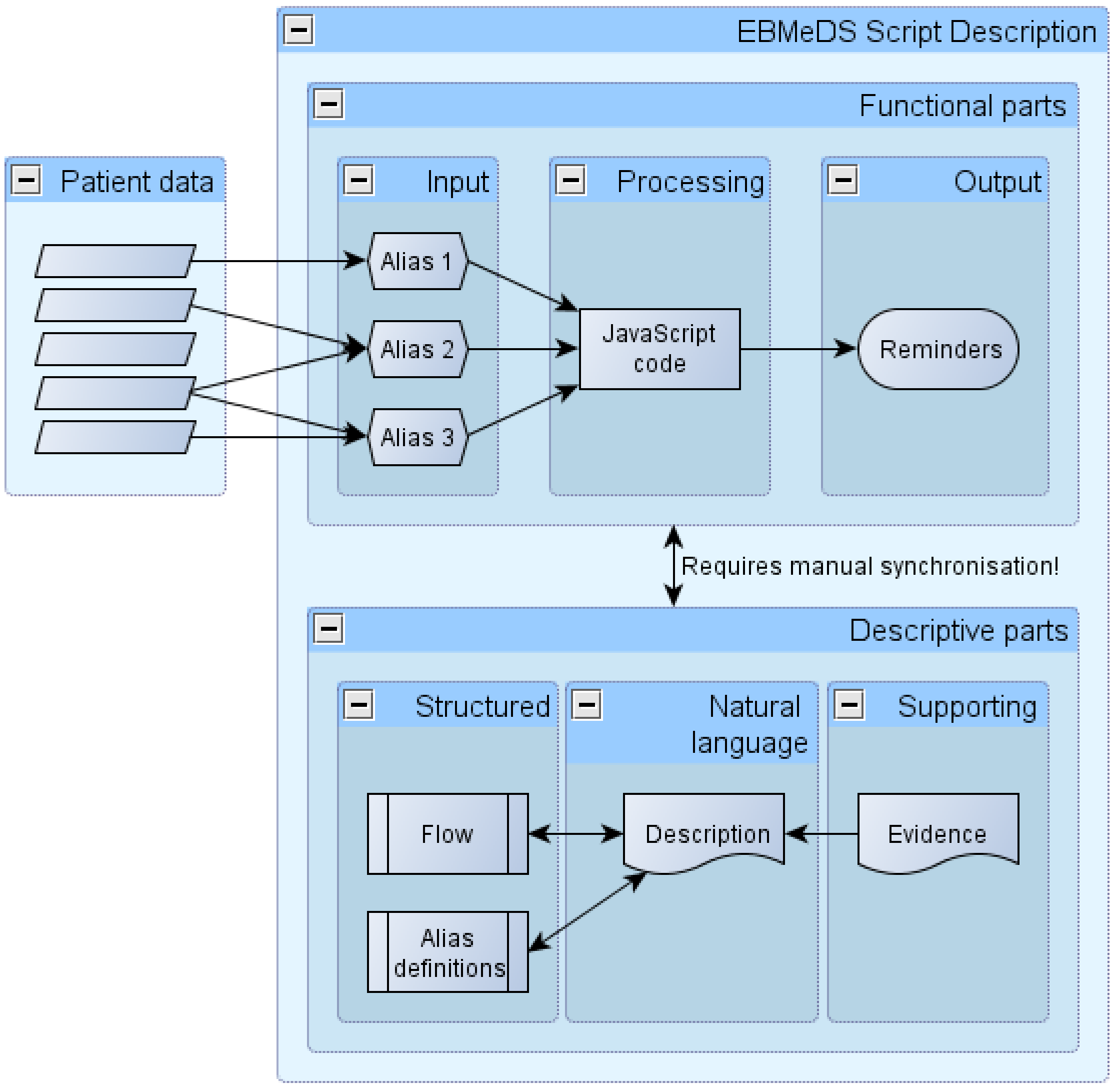
I. Refining the script idea into a script description (Description Stage)

The purpose of the description stage is to reconcile the script idea with limitations imposed by the EBMeDS framework and by the real-world availability and reliability of the data the script needs to function. Because nascent script ideas are usually more solution than implementation-oriented, they tend to need at least further detailing with respect to technical implementation (e.g. specific cut-off values guiding script action need to be chosen for clinical variables and their temporal relationships). Mostly the ideas will also need to be altered to some extent to satisfy requirements imposed by the EBMeDS framework and the availability and reliability of structured patient data. If a script idea cannot be implemented with sufficient quality due to these boundary conditions, it's development can either be paused pending favourable developments in boundary conditions (e.g. better availability of structured patient data), or discontinued altogether.

The end product of the description stage is a detailed *script description*, edited through the *EBMeDS Script Description Editor (ESDE)* and stored within the EBMeDS Central Database. It should contain all the information necessary to understand, assess, justify and implement the script. Clarity and consistency of the script description are vital for later stages of the development process and should receive special attention: the script description consists of distinct parts that nevertheless have some overlap in the information or functionality that they describe, making it relatively easy to create inconsistencies if care is not taken.

An important part of upholding the consistency of the script description is keeping its descriptive and functional parts synchronous. As a rough generalisation the descriptive parts explain the script for human readers while the functional parts describe it for computers. Changes to the functional parts directly affect script functionality, whereas changes to the descriptive parts do not. Figure 2 below illustrates the distinction between these parts.

Figure 2: The functional and the descriptive parts of the script description



The parts of the script description which carry most importance with respect to enabling the continued high-quality implementation of the script idea are:

- a plain language *description* of the script's functionality;
- references to *evidence* justifying the functionality;
- *alias* definitions, the script's main channel of access to patient data; and
- *reminders*, the script's only channel of communication to the end user

The importance of the script description can be understood in light of the fact that script development and maintenance are highly collaborative multi-disciplinary tasks that depend on clear and accurate exchange of information among experts. Every script implementation will require at least medical and technical expertise. Implementing scripts with good quality usually also requires detailed insight into the target environment (e.g. care pathways and other processes used at the provider organisation; work-flow patterns of health-care professionals; regulatory requirements; etc.). Thus the role of the script description as the repository of all information pertaining to the script makes it pivotal in determining script quality. It should fulfil the following information functions:

1. convey the information necessary for implementation to the programmer;

2. serve as a basis for various peer-review processes, both pre-publication internal reviews and external reviews by independent experts and other interested parties;
3. enable the end-user to assess the importance and applicability of the script's feedback with respect to the patient and the current context in general;
4. act as a facade for the script and the entire EBMeDS system for end-users, present and potential partners, and other stakeholders

Due to the importance of clarity and consistency of the script description, unnecessary scattering of the work that goes into it, either across time or human resources, should be avoided. However, the script ideas that serve as the starting point of the description stage should be easy to submit for anybody regardless of technical familiarity with the EBMeDS framework. As the idea is refined further, sufficient expertise on medical, technical, organisational, and other relevant aspects should be available.

II. Technical implementation of the script description (Programming Stage)

The aim of the programming stage is to implement the technical aspects of the script according to the plan made during the description stage. The functional logic described in the script description in human-readable form is translated into JavaScript code that can be executed by computers. From a quality perspective, it is decisively important that the programmer understands the desired functionality correctly. If the script description is clear, detailed and unambiguous, the programming stage can be a very quick and straightforward operation, with an outcome of predictable quality. In the opposite case misconceptions can threaten script quality in unpredictable ways.

In practice, the programming and description stages are seldom as completely independent from each other as depicted above. It often becomes necessary to refine or modify some aspects of the script description as the programming stage progresses. It is therefore important that the programmer and the script description author(s) can consult and collaborate with each other when necessary. For this same reason the two stages should benefit from being timed closely together. When refinements and modifications are made, it is important to uphold the consistency of the script description by keeping its functional (JavaScript code, reminders, aliases) and descriptive (all other) parts synchronised.

The end product of the programming stage is a small program snippet – or a script – written in the JavaScript programming language, which together with the aliases and reminders ultimately determines how the script behaves in a given context. In contrast, the rest of the script data editable through the ESDE does not at present affect script functionality, but merely aims to describe it (see Figure 2).

From a quality standpoint the program code should implement the intended functionality flawlessly and require minimal resource expenditure when maintenance is inevitably needed. Both goals are served by short and clear code which conforms to mutually agreed-upon structure and notation. A central tenet is building code around EBMeDS proprietary library methods, which enables individual scripts to be written in simple, self-explanatory form, while more complex tasks are taken care of by library functions. This enables trained health care professionals with little earlier programming experience to serve as script programmers.

III. Peer review of script description and implementation (Testing Stage)

The testing stage can be seen as a peer review process within the EBMeDS Editorial Team, with the purpose of ensuring that the script is of sufficient quality for publication. The review process focuses on the script description prepared during the description stage on the one hand, and on the technical implementation built upon that description on the other hand.

The review will focus on, in particular:

- clarity, consistency, and completeness of the script description
- expected costs versus benefits from the described functionality
- correspondence of intended and actual functionality
- meaningfulness of the functionality as a part of the whole EBMeDS system

External expert opinions are used in the review process if the EBMeDS Editorial Team deems it appropriate. In such cases opinions can be requested e.g. from guideline developers, experts of appropriate medical fields, other health professionals, public authorities, physician or patient groups, or other applicable sources.

Technical implementation is tested with the web based EBMeDS Test Application. Its basic function is to allow feeding of made-up patient data to the decision support engine in order to find out what feedback is produced.

Simulation of EBMeDS output in a larger patient population is feasible via “a virtual health check”, where all scripts are executed as a batch run for a representative sample of the target organisation’s patients. The anonymised log files of the batch run will contain information on numbers of reminders elicited, and quality measure statistics based on the scripts. Unexpected results in the virtual health check may reveal problems in patient data or script logic.

Testing technical implementation also entails peer review of the actual JavaScript code of the script by other EBMeDS script programmers.

The final decision to publish rests with the EBMeDS Editor-in-Chief. After the script has been reviewed by at least three members of the EBMeDS Editorial Team, the Development Director may offer it to the Editor-in-Chief for publication. If the editorial group has deemed it appropriate to request assessments by external experts, the Editor-in-Chief should consider them when deliberating publication. The publication decision should be recorded in the notes of the script editor, together with the rationale behind it.

IV. Post-publication maintenance (Maintenance Phase)

The maintenance phase aims to maintain the quality of published scripts at a high level over time. Maintenance needs arise at least through the following two mechanisms:

1. a mismatch between the script and its specific operating environment can produce unexpected and undesirable behaviour
    a. when a script is first commissioned in an incompletely understood real-world operating environment
    b. when the operating environment changes e.g. through constantly evolving workflow patterns, software, coding standards, regulatory requirements etc.;
2. evolving medical knowledge can render scripts obsolete

Central challenges for successful maintenance of script quality will be detecting the need for maintenance, along with choosing the most appropriate maintenance action and deciding its priority.

Sufficient contact with end-users and organisations using EBMeDS is required to efficiently detect and deal with problems that arise from mismatches between scripts and their operating environment (item 1 above). Even after careful planning and testing, a script may function unexpectedly in a real-world environment, as comprehensive pre-publication testing of script function in realistic circumstances is not currently feasible. This is partly due to the protected nature of patient data, and partly due to the great variability of operating environments that EBMeDS may be deployed in (this includes software and hardware, but also workflow variability on both organisational and individual levels).

Systematic monitoring of evidence and clinical practice guidelines linked to the script description is key to detecting maintenance needs due to evolving medical knowledge (item 2 above). Periodic reviews of evidence and various services that aim to summarise important developments in medical knowledge may be utilised for this task. Related tasks are also described in detail in the documentation for the EBM Guidelines Database.

To the extent that maintenance needs of an individual script can be anticipated, they should be taken into account when devising the script's individual maintenance plan which is stored in the script description.

After a need for maintenance is identified, appropriate measures and their urgency should be decided upon. Withdrawal of a script from production may at times be more appropriate than trying to update it; this could be the case, for example, when best practices have radically changed, or a new script covers the same functionality. Generally speaking, a script should be withdrawn if it has a serious problem that cannot be solved, or if working around it would require unreasonable efforts in relation to the achieved benefits. The urgency of maintenance action is decided based upon the gravity of the problem.

General quality affecting processes

Development and maintenance of the EBMeDS system involves quality affecting parts that are not directly related to the production processes of individual scripts. Examples include collecting information on user experiences, development of the EBMeDS framework and function libraries, and proactively monitoring quality affecting developments in the environment that EBMeDS operates in.

Communication with end-users

Sufficient contact with end-users is an integral part of quality management. Firstly, patient data confidentiality and the diversity of possible operating environments for EBMeDS make it difficult to comprehensively test the system in realistic conditions. This means that unanticipated problems will probably manifest themselves to end-users after commissioning EBMeDS or a specific script into use in a new environment. Secondly, the large number of EBMeDS users and their patient contacts mean that potentially great amounts of useful quality-related information flows by the end-users. In order to tap this information, a fast-and-easy feedback channel is available from the end-users to the EBMeDS developers. The EBMeDS team should also regularly initiate contact with end-users. The resulting feedback should be carefully documented and analysed on a regular basis to make it useful in improving the quality of the EBMeDS system.

Monitoring maintenance needs

Active monitoring of developments that necessitate updating how EBMeDS functions is a prerequisite for maintaining its quality. In addition to the EBMeDS Editorial Team, this task is carried out by the EBM Guidelines Editorial Team in Finland, and by script-producing organisations in other countries. Maintenance needs are caused by e.g.

- the continuous expansion and refinement of medical knowledge and treatment possibilities;
- organisational, legislative, or bureaucratic changes affecting how health care services are provided on all levels, ranging from personal workflow patterns of health care professionals to the structure of the larger scale health care system; and
- changes in those information and classification systems that EBMeDS must be compatible with in order to function correctly

Developing the EBMeDS framework

A key strength of the EBMeDS system lies in its simple and flexible structure, which allows health care professionals to be closely involved in the management and development of decision support functionalities without extensive training in information technology. As the quantity and quality of structured patient information in EHR's grows, so grows the potential of EBMeDS to provide useful functionalities to its users. With this in mind, the EBMeDS framework should be continuously developed with the aim of enabling clinically relevant functions to be implemented in a simple and clinically relevant manner at the script level.

Influencing external boundary conditions for quality

Many external factors set boundaries to the kind of functionalities that can be implemented with high quality via EBMeDS. Quite a few of these factors produce their effect by influencing the quality and quantity of structured patient data available to EBMeDS. Ease of use and descriptive power (or resolution) of existing coding systems could be named as examples.

As availability and reliability of clinical data ultimately limits the potential of any clinical decision support solution, it would make sense to influence these external factors when and where ever possible. The mere existence of EBMeDS and other CDS solutions will hopefully encourage more diligent recording of structured patient data, but active co-operation with standards organisations, and even development and deployment of useful structured data formats in collaboration with partners for the purposes of accelerating wider utilisation of useful data in CDS settings, is to be encouraged.

Measures for improving quality

The following sections aim to present in condensed form the concrete measures relating to the production and maintenance of script functionalities that aim to ensure the highest possible quality of the EBMeDS system. The measures have been divided according to whether they relate mainly to the production process of a (single) script, or to EBMeDS development and maintenance processes on a more general level. Figure 3 at the end of the section seeks to illustrate and summarise the information presented. Parts in grey text refer to measures which have been suggested but not yet implemented.

Measures relating to the script production process

Table 1: Measures relating to the Description Stage (existing and planned)

Clear description	• **Structured**: describes in layman's terms the script's *aim*, *context* (including the targeted patient group), and *methods*. • **Detailed enough**: sufficient detail to enable end-users (including non-professionals) to understand why a specific reminder was triggered and what evidence supports it; still concise and manageable (quality over quantity) • **Up-to-date**: updated whenever any part of the script is changed (see Figure 2)!
Evidence	• Is the evidence consistent with the script's *aim* and *methods*? • Is the evidence valid in the script's *context*? • Could the script context be more closely matched to the evidence?
Short reminder	• Plain and simple form, e.g. "Triggering reason – recommended action" • Many end-users may only ever look at the short reminder: they should contain **enough information** on their own to be useful in guiding clinical action.
Long reminder	• The triggering reason and the recommended action may be presented in more detail than in the short reminder. • The aim is to minimise the extra effort required to assess the reliability and applicability of the given information: ○ directly actionable (clear, unambiguous) recommendation
Keywords	• Systematic indexing of scripts with keywords (tags) – based on controlled vocabulary (e.g. MeSH) where applicable – according to relevant features, with the aim of facilitating ○ compatibility and redundancy checking across scripts ○ management and development of script families relating to specific themes or clinical areas ○ identifying (or flagging) scripts in need of maintenance ○ conceptual grouping of scripts e.g. for the purposes of EBMeDS-related research or presentation to interested parties
Compatibility checking	• Compatibility with other scripts aimed at similar clinical contexts? • Could these scripts be combined or harmonised otherwise? • Does the evidence supporting the scripts allow harmonisation?
Flow	• If the Flow-syntax proves ill-suited to describe script function, don't use it – prioritise a clear description instead. • The Flow-part only serves to describe a script's functional logic – changes to it do not change actual script function!
Aliases	• Sketching up aliases already during the description stage helps detecting technical and data-quality issues that may limit quality • Avoid duplicate aliases • If necessary, create new aliases. Tools to find the appropriate codes can be found in the Script Editor. • See also a separate document on the purpose and use of aliases
Potential Harms	• Systematic reflection and documentation of potential harms on two axes: ○ harms related to the medical adverse effects and potential complications of the recommended actions ○ harms related to difficulties in correctly targeting reminders (high proportion of unnecessary or harmful reminders) • Not to be taken as an exhaustive listing of all possible harms, but

	rather an educated guess of the script producers which may be amended when experience of the script accumulates
Maintenance	• A Maintenance part will be added to the Script Editor, containing o a maintenance plan for the script (originally drawn up during the description stage and updated later when necessary) o a last assessed as up-to-date timestamp o a log of changes or updates • The maintenance plan includes a checklist of triggering events that should initiate an assessment of maintenance needs for the script. Triggering events may include: o predetermined dates (e.g. a Cochrane update or an EBMeDS Editorial meeting) o updates affecting entries in locally produced and maintained (by Duodecim Medical Publications Ltd) evidence databases that are linked to the script o user feedback relating to the script o changes to the coded classifications utilised by the script o external triggering events independent of Duodecim are harder to track with limited resources ▪ Automatic alerting services that monitor changes in specific databases or web-sites (e.g. Google Alerts) may be useful ▪ For scripts that have been tailored to specific local circumstances, tracking of trigger events should be taken to the local level to be effective

Table 2: Measures relating to the Programming Stage

Script description	• **Keep functional and descriptive parts synchronised** as technical implementation takes on its final form (see Figure 2) • **Low threshold to co-operate** with the author of the script description if the programmer finds the script description unclear
Code clarity	• The program code should be as clear and short as possible o mutually agreed-upon basic code structure and notation. • If complex or cumbersome code becomes necessary in a script: o report the problem (see Developing the EBMeDS framework) o use ample commenting as a short term solution o over the long term, the programming team will aim to develop library functions and other parts of the EBMeDS framework to support simple and flexible implementation of useful functionalities at script level (for a more detailed description, see Developing the EBMeDS framework)
Testing by the programmer	• The programmer should always carefully test the function of the finished script. The programmer's unique technical knowledge of the script's structure and the challenges that were encountered during implementation allows focusing of testing efforts on those parts of the script's function that may be particularly susceptible to errors.

Table 3: Measures relating to the Testing Stage (existing and planned)

Internal (EBMeDS editorial) peer review of the Script Description	• Does the Script Description confer a clear, unambiguous and justified picture of the script's function? The following parts should be closest scrutinised: o Description

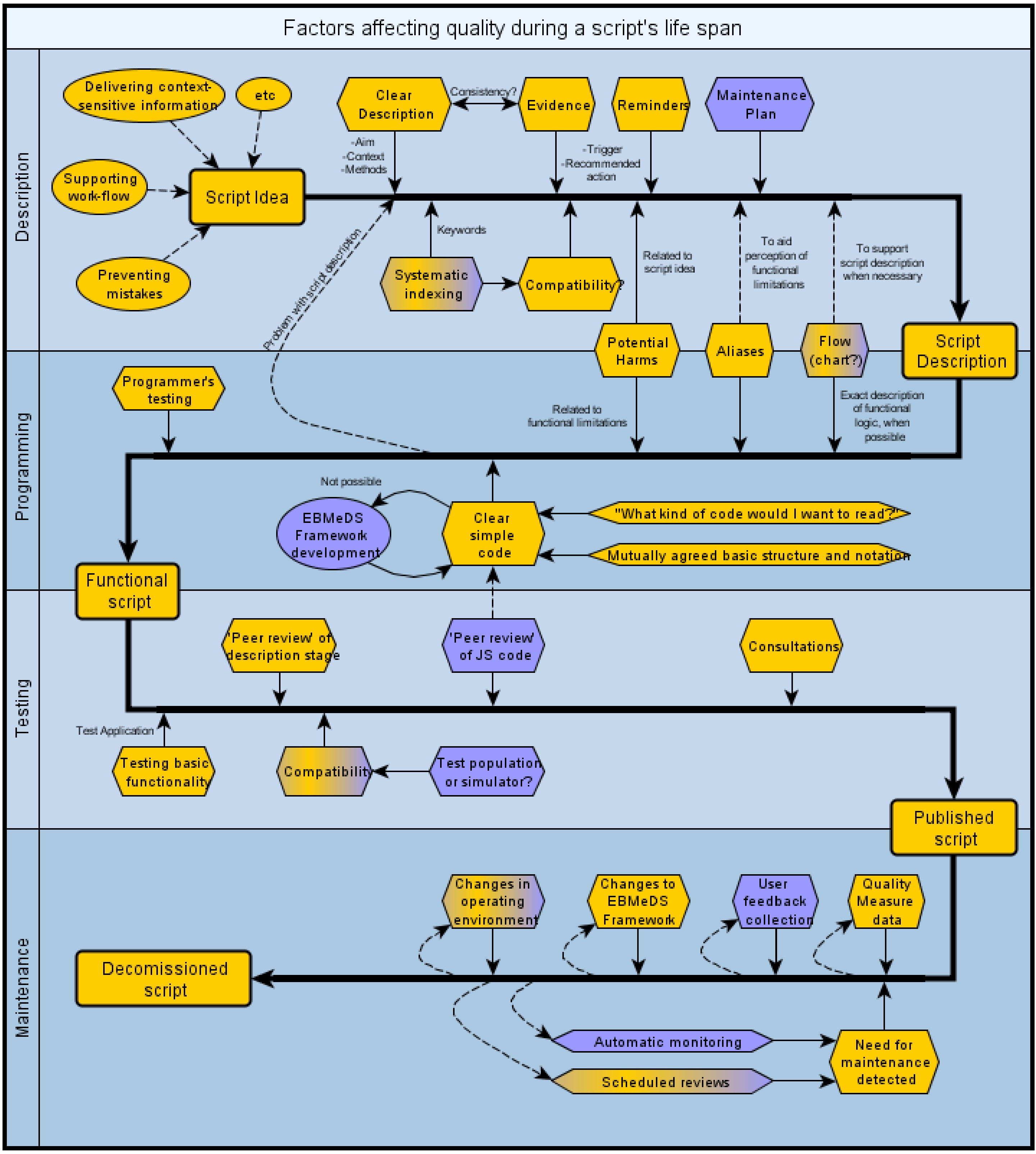
	○ Evidence ○ Use of structured data (Aliases) ○ Reminder messages ○ Overall consistency • Does the Script Description contain sufficient information, especially in relation to needs of external stakeholders and interested parties? ○ Understanding the script's intended function and forming an opinion as to its usefulness ○ Troubleshooting errors • Should external experts or stakeholders be consulted before publication of the script?
Review of script function	• Does actual script function (assessed by the web based EBMeDS Test Application) correspond to the Script Description? • A future aim is to enhance testing by building a realistic test population, based on anonymised real patient data. This would narrow the gap between testing and real-life circumstances, and could possibly increase the number of hidden quality problems detected before publication.
Peer review of JavaScript code	• Light, informal peer review process, with the aim of ○ receiving feedback on your own code ○ learning from other programmers' coding ○ being less unfamiliar with another programmer's code if it becomes necessary to update it ○ early detection of the need for updating the EBMeDS Function Library

Table 4: Measures relating to the Maintenance Phase (existing and planned)

Systematic analysis of Quality Measure data	• Quality data produced by the scripts (Quality Measure, QM) is analysed regularly • What are the reasons for low QM scores? ○ Is the script generating unnecessary reminders due to unnecessarily loose context recognition? ○ Are users ignoring reminders because their content is ▪ obsolete or wrong? ▪ too difficult to apply or assess?
Script specific user feedback collection	• Implementation examples: ○ Collaboration with EHR suppliers to build a user feedback function directly into the EHR's user interface ○ Enabling both a quick thumbs up/down type of feedback and a free text type of feedback ○ A feedback link at the top of each script's web page at the EBMeDS home pages (visible immediately upon entering the web page via e.g. a link in the reminder text) ○ Let the user specify an e-mail address if he/she wishes to receive a reply from the EBMeDS Editorial Team • See also notes on general feedback concerning the EBMeDS system
Monitoring the need for maintenance or removal	• Follow the maintenance plan of the script • Changes in evidence and other factors that create maintenance needs can be detected and remedied (see also notes on trigger events in the maintenance plan) ○ by regular checking (e.g. Cochrane publishes its update

	schedule) ◦ as a part of the update process of script-linked entries in evidence databases managed in-house by Duodecim (e.g. Evidence Reviews, EBM Guidelines, Current Care guidelines) ◦ by user feedback (good cost-benefit ratio) • After a need to update or remove a script is detected, resolve ◦ urgency of maintenance ◦ most appropriate method of maintenance

Figure 3: Factors affecting quality during a script's life span. Colour coding: orange – existing part of the script production process; purple – proposed addition; gradient – existing part with proposed amendment



Factors affecting quality during a script's life span
Description
Programming
Testing
Maintenance
Delivering context-sensitive information
etc
Supporting work-flow
Preventing mistakes
Script Idea
Clear Description
-Aim
-Context
-Methods
Consistency?
Evidence
-Trigger
-Recommended action
Reminders
Maintenance Plan
Problem with script description
Keywords
Systematic indexing
Compatibility?
Related to script idea
To aid perception of functional limitations
To support script description when necessary
Potential Harms
Aliases
Flow (chart?)
Script Description
Programmer's testing
Related to functional limitations
Exact description of functional logic, when possible
Not possible
EBMeDS Framework development
Clear simple code
"What kind of code would I want to read?"
Mutually agreed basic structure and notation
Functional script
'Peer review' of description stage
'Peer review' of JS code
Consultations
Test Application
Testing basic functionality
Compatibility
Test population or simulator?
Published script
Changes in operating environment
Changes to EBMeDS Framework
User feedback collection
Quality Measure data
Decomissioned script
Automatic monitoring
Scheduled reviews
Need for maintenance detected

Measures relating to general (not script-related) processes

Table 5: Measures of general nature (existing and planned)

Contact with stakeholders and interested parties	• User Feedback System (general feedback) o Web site feedback form o E-mail (e.g. feedback@ebmeds.org) o Opportunity to request reply to specified e-mail address o Feedback through related or integrated services managed by Duodecim (e.g. EBM Guidelines) o See also script specific feedback • Active user feedback collection after rolling out EBMeDS at a new location o User questionnaire after a few months of use? o Actively inquire about user experiences whenever in contact • Educate, make information available, encourage social networking o Clear and informative EBMeDS-site ▪ Enable keyword search as in the ESDE ▪ Build features supporting social networking • Script specific public comments or discussion? • Script specific public voting (e.g. +1/-1) and ranking-list? o Mailing-lists o Feeds (RSS, Atom, etc.) o Instant message networks / Social Media ▪ Twitter, Clinical Cafe ▪ Presence at general professional discussion boards? ▪ Build own discussion boards at the EBMeDS home page?
Staying current on maintenance needs	• Monitor changes in classifications that code structured patient data, or in legislation or bureaucratic requirements that affect clinical work-flow o Good links to standards bodies and governmental institutions • Periodic review of evidence • Automated alert services • See also script specific monitoring of maintenance needs
Developing the EBMeDS Framework and Tools	• The EBMeDS Framework and function libraries o A facility for reporting problems related to the EBMeDS Framework should be built into the ESDE. A script programmer could use it to report a problem interfering with code clarity, and propose a solution ▪ Various collaborative tools could also be considered as a platform for reporting and discussing problems, deciding how to solve them, and documenting the whole process o A development team should regularly review the reported problems and ▪ decide on the urgency of remedying the problems ▪ seek to find solutions by developing library functions and other parts of the EBMeDS Framework • The ESDE, e.g. o Automating alias name changes in affected JavaScript code? o Other types of changes (i.e. content changes or removal of an alias) could use e.g. keywords to flag the scripts that use the affected aliases? • Interface development, e.g.

	◦ When coded classifications change ◦ For promoting the use and utilisation of specific structured data • The EBMeDS Test Application ◦ building a realistic test population
Influencing external boundary conditions	• Co-operation with standards organisations • Offering proprietary structured data formats to promote use and utilisation of useful structured data

Glossary

***Aim* (script)**	What the script aims to accomplish or prevent. The main reason of a script's existence, and its justification. The aim should be explicitly stated in the *description* field of the *Script Description*, in layman's terms if possible.
Alias	A descriptive name by which the *script* refers to structured patient data it needs to interact with. They consist of a group of codes from relevant coded classifications (e.g. ICD10, ATC, LOINC, etc.). There is a separate document describing aliases.
***Context* (script)**	In a broad sense, the entire environment in which the *script is* planned to operate. This includes intended user groups, targeted patient groups, the subset of the health care system in which the script is intended to be used (e.g. primary health care, maternity clinics, occupational health care, cardiology clinic, a specific geographical or organisational block, etc.). The main features of the intended context should be stated explicitly in the *description* field of the *Script Description.*
***Description* (plain text)**	A field of the *Script Description*, the contents of which serve to provide a natural language description of the scripts functionality in layman's terms. The description should state explicitly the script's *aim*, *methods*, and *context*. Not to be confused with the *Script Description*, of which it is a part.
Evidence	A section of the Script Description which presents short summaries and direct links to evidence and guidelines that the script's idea and functionality are based upon. It thereby supports and complements the description in justifying the usefulness and legitimacy of the script, and should therefore be consistent with the script's aim, methods and context.
Flow	A structured section of the Script Description that aims to clarify the script’s functional logic for a human reader. It offers a simple row-by-row syntax which is adequate to represent many of the simpler scripts, but it doesn’t work well with the convoluted logic found in some more complex scripts. The flow is purely descriptive, and has no direct bearing on actual script function.
Keywords	A field of the Script Description containing keywords (or tags) that can be used to classify the script in multiple relevant categories, allowing it to be found more easily, and facilitating various development and management activities.
***Methods* (script)**	A script's functional logic, i.e. how it strives to accomplish its *aim*, taking into account limitations affecting implementation. The methods should be stated explicitly in the *description* field of the *Script Description*, in layman's terms if possible, but still in enough detail to be useful for end-user troubleshooting e.g. after receiving a seemingly unwarranted *reminder*.
***Quality* (script)**	The concept of script quality is defined as the increase in quality of the entire EBMeDS system that a specific script produces. The notion of quality has been defined in a separate document.
Reminder	A message from the *script* to the user. Its purpose is to provide the user with

	useful context sensitive information to support clinical decision making and efficient workflow execution. A more detailed discussion can be found in a separate document.
Script	A component of the EBMeDS system which implements a specific, independent functionality, based on a *script idea*. A script works by analysing patient data and showing reminders to the user based on that analysis. The word "script" is used in EBMeDS's context to refer not only to the JavaScript-code implementing the script's functional logic, but to the entire collection of information relating to the script (including *description*, *evidence*, *aliases* and *reminders*), which is stored together as a *Script Description* in the EBMeDS *Script Description Editor* (ESDE).
Script Description	The centralised repository and mediator of all information related to a specific script. It serves the information needs of both the script development team during the pre-publication production process and any external interested parties after publication. Not to be confused with the (plain text) *description* (see below), which is one part of the *Script Description*
Script Editor, ESDE	A comprehensive web-based tool for developing and maintaining scripts. Contains useful search tools and allows editing all information contained in the *Script Description*. ESDE = EBMeDS Script Description Editor.
Script idea	The first rough sketch of a functionality that is proposed to be added to EBMeDS. It is usually concentrated around describing the script's *aim*, while *methods* and *context* receive less focus.