

Processing of representations in declarative and procedural working memory

Alessandra da Silva Souza, Klaus Oberauer, Miriam Gade, and Michel D. Druey

Department of Psychology, Cognitive Psychology Unit, University of Zurich, Zurich, Switzerland

The article investigates the relation between declarative and procedural working memory (WM; Oberauer, 2009). Two experiments test the assumption that representations in the two subsystems are selected for processing in analogous ways. Participants carried out a series of decisions on memorized lists of digits. For each decision, they had to select declarative and procedural representations. Regarding declarative representations, participants selected a memory set and a digit within this set as the input to each decision. With respect to the procedural representations, they selected a task set to be applied to the selected digit and a response within that task set. We independently manipulated the number of lists and the number of tasks to be switched among (one, two, or three; Experiment 1) and preparation time for a list switch (Experiment 2). For three effects commonly observed in task-switch studies, analogues in declarative WM were found: list-switch costs, mixing costs, and residual switch costs. List- and task-switch costs were underadditive, suggesting that declarative and procedural representations are selected separately and in parallel. The findings support the hypothesis of two analogous WM subsystems.

Keywords: Working memory; Task switching.

Activities such as mental calculation, reasoning, or reading comprehension are considered as instances of complex cognition. Complex cognitive tasks have two characteristics in common: (a) They require that some information is made temporally available, while (b) some cognitive processing is performed on this information. Working memory (WM) is considered the system underpinning both of these functions—namely, temporary storage and processing (e.g., Baddeley, 1986; Miyake & Shah, 1999).

Most research on working memory so far has focused on the storage function of WM—for instance, by examining how much information people could maintain for short periods of time in the face of some distracting activity, as in the complex memory span tasks (e.g., Daneman & Carpenter, 1980). At the same time, other researchers studied cognitive processes when several tasks have to be performed in close succession or simultaneously, using task-switching or dual-task paradigms (e.g., Monsell, 2003; Pashler,

Correspondence should be addressed to Alessandra Souza, University of Zürich, Department of Psychology, Allgemeine Psychologie (Kognition), Binzmühlestrasse 14/22, 8050 Zürich, Switzerland. E-mail: a.souza@psychologie.uzh.ch

This work was funded by a grant from the Swiss National Science Foundation (SNSF) to Miriam Gade, Klaus Oberauer, and Michel Druey (Project 130.113).

1994). Whereas the first line of research uncovered limits on how much information can be maintained as available for processing, the second line revealed limitations on how much information can be processed at the same time.

These two types of cognitive limit have been rarely interpreted within one unified WM model. In an attempt to sketch a more comprehensive view of the WM system, Oberauer (2009) proposed a framework in which the representations of the to-be-manipulated information as well as the representations of the different (mental) operations that can be applied to this information are maintained separately in two WM subsystems: declarative WM and procedural WM. The function of both subsystems is to hold available the small set of representations that are temporarily selected to guide our thoughts and actions. Representations of entities in the world (e.g., objects, events, words, numbers) are the domain of the declarative subsystem, whereas representations of condition–action rules—such as “if the number is even, then press the left button”—are maintained as available by the procedural subsystem.

The main purpose of this study is to test the hypothesis that analogous mechanisms serve to select declarative and procedural representations. We used the model proposed by Oberauer (2009) as a theoretical framework to test whether selection of these two types of representation are accomplished by similar means in WM, so that analogous experimental manipulations—intended to bear either on declarative or procedural WM—would produce analogous, but independent, behavioural effects across these subsystems. In the following, we describe in more detail the model of declarative and procedural WM along with current evidence for it. Next, we elaborate the goals pursued in the present experiments and the set of predictions derived from the assumption of two WM subsystems that operate in analogous ways.

The design of the declarative and procedural WM subsystems

According to Oberauer (2009), WM generally serves the selection of representations for goal-

directed processing. This function is accomplished by simultaneously determining the relevant memory contents (declarative) and the relevant processing operations (procedural) necessary to accomplish a cognitive goal. Declarative working memory (dWM) and procedural working memory (pWM) subsystems are assumed to have analogous structures. Both consist of three embedded components, which, in functional terms, could be described as three states of representations as they become selected for goal-directed processing. The general architecture of dWM and pWM is schematically illustrated in Figure 1.

The first component of both subsystems is called the *activated part of long-term memory* (LTM). Representations in LTM are assumed to be activated above baseline by perceptual input or the spread of activation from other representations (Anderson & Lebiere, 1998). Declarative and procedural representations are activated simultaneously in LTM and activate each other through associations, such that no clear separation is assumed at this level. Activation constitutes the first level of selection of relevant representations: The level of activation of a representation provides a quickly available, automatic index of its likely relevance for the upcoming situation or task. However, activation spreads unconstrained along all associations, such that many irrelevant and potentially misleading representations are also coactivated along with relevant ones. Therefore, the role of the second level of selection is to single out a small set of goal-relevant representations to be maintained in a more accessible state for further cognitive processing.

This set of goal-relevant representations constitutes the contents of the *region of direct access* (DA region) on the declarative side, and the *bridge* on the procedural side. For example, when participants have to memorize a list of words for later recall in order, each word triggers the activation of many other related words (as well as some procedures associated to them), but only list words are maintained in the DA region. Moreover, the DA region provides temporary bindings of each word to its position within the list. Likewise, when participants must perform a task (e.g., categorize

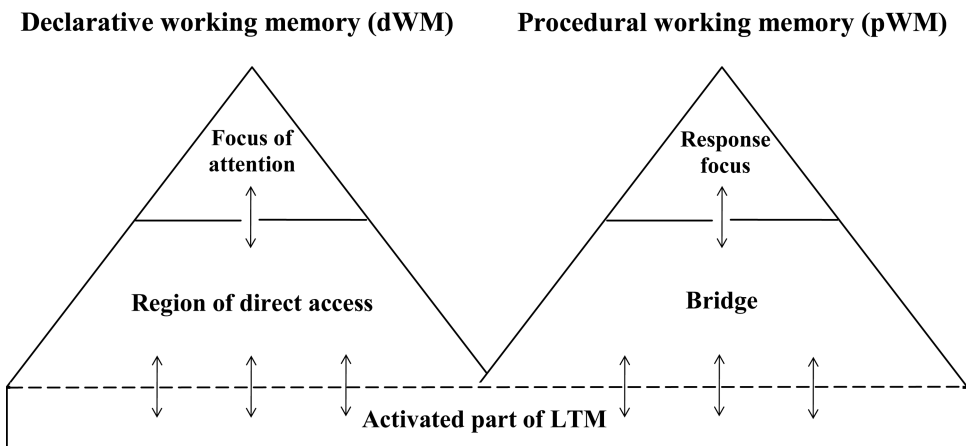


Figure 1. Architecture of the declarative (dWM) and procedural working memory (pWM) subsystems. Each box represents one level of a three-levelled embedded structure (e.g., activated part of long-term memory, LTM; bridge; and response focus). The position within the pyramid indicates both the accessibility of representations in that component and the component's limits. Therefore, the component depicted on the top has the smallest capacity to hold information, but contains the most available representation.

printed words according to their colour), only the relevant classification and its condition–action rules (e.g., “if the word is printed in red, press the left button; if the word is printed in blue, press the right button”) should be kept in the bridge to the exclusion of irrelevant tasks that are activated in LTM because of their association with the stimuli (e.g., “read the word”). With the help of temporary bindings between stimulus categories and response representations, any arbitrary stimulus–response mapping can be configured rapidly in the bridge, thus enabling the system to overcome well-learned habits and to rapidly switch between different responses to the same stimuli.

The distinction between the activated portion of LTM and the more limited central components of the declarative and procedural WM (i.e., the DA region and the bridge, respectively) has been corroborated by several findings. When people encode two lists of digits or words for a test of immediate memory, for instance, and then are cued as to which list is relevant for the test, they quickly discard the other, irrelevant list from the DA region. This is reflected in the differential set-size effects of both lists on the reaction times (RTs) for responses to the test stimulus (e.g., a recognition probe): Whereas the relevant list generates a robust

set-size effect, the set-size effect of the irrelevant list decreases with increasing time between the cue and the test stimulus, vanishing after 1–2 s (Oberauer, 2001, 2002, 2005). When the list initially declared as irrelevant is cued as relevant on a second test, that list can quickly be retrieved back into the DA region, as shown by the reemergence of set-size effects of that list and people’s high accuracy on a memory test for that list (Oberauer, 2005). Switching between the two lists for successive tests, however, takes time compared to repeatedly testing the same list (Oberauer, 2005). This switch cost has been interpreted as further evidence that only the currently relevant list is maintained in the capacity-limited DA region, while currently irrelevant information stays in the background—that is, in the activated portion of LTM.

Concerning procedural WM, its three hypothetical components have rarely been addressed explicitly in the literature so far. Nevertheless, evidence for the distinction between its components can be drawn from simple choice tasks and task-switch studies. In typical task-switch experiments, for instance, participants have to perform one of two tasks on stimuli that afford both tasks (e.g., decide whether a digit is smaller or larger than five or

whether it is even or odd). Moreover, the tasks appear either in random order (prespecified by a task cue in each trial, e.g., Meiran, 1996) or in a predefined order (e.g., alternating runs, Rogers & Monsell, 1995). In both cases task-repetition and task-switch trials can be compared, and the general finding is that responding in task-switch trials is slower than that in task repetition trials (Altmann, 2004; Meiran, 1996; Rogers & Monsell, 1995). This finding suggests that whenever a task switch has to be carried out, the old task has to be removed from the bridge, while the new task is retrieved from activated LTM. Since this takes some time, performance in task-switch trials is impaired relative to task repetition trials, thus incurring task-switch costs (Mayr & Kliegl, 2000).

Further evidence comes from the so-called mixing costs. Mixing costs reflect the slowing of responses in repetition trials of mixed-task blocks (in which two or more tasks have been instructed to be potentially relevant) compared to a single-task block (in which only one task is instructed). At first glance, mixing costs could be interpreted as reflecting the higher load on the bridge—which is assumed to have limited capacity like the DA region—that is imposed in mixed-task blocks than in single-task blocks. However, if mixing costs reflected a load effect, they should increase with the number of tasks in mixed-task blocks. Contrary to this expectation, some researchers found that mixing costs do not increase further if the number of tasks in a mixed block is increased beyond two (Rubin & Meiran, 2005; Steinhäuser & Hübner, 2005). This finding suggests that not all tasks involved in a mixed-task block are held in the bridge simultaneously; rather, the task sets not relevant for the current trial are likely to be held in the more unconstrained activated portion of LTM. Thus, switching to a new task involves removing the old task set from the bridge and retrieving the new task set from activated LTM (Mayr & Kliegl, 2000). Furthermore, this assumption is supported by evidence showing that the influence of currently not selected task sets (such as the task-congruency effect) reflects their residual activation in long-term memory (Kiesel, Wendt, & Peters, 2007; Meiran & Kessler, 2008).

The third level of selection singles out individual representations. After the selection of the subset of relevant declarative representations in the DA region, the system still needs to select one item as the object of manipulation in dWM. Likewise, after the selection of a task set in the bridge, the system still needs to select one response to be executed in pWM. Therefore, the third component of each subsystem is an *attentional focus*, which serves to single out the currently relevant target object on the declarative side and the selected response to this object on the procedural side. For example, if the task is to recall the third word of a list, cueing with the list position would bring it to the focus of attention. If the same word is repeatedly cued by the same position, response times to the target word are faster than when a switch is required to a different word in a different position of the same list. Evidence for these object-switch costs has been obtained in n-back tasks (Oberauer, 2006) and several variants of the memory-updating tasks, such as the running-counters task of Garavan (1998), or the arithmetic (Kessler & Meiran, 2006; Oberauer, 2002, 2003) and spatial (Kübler, Murphy, Kaufman, Stein, & Garavan, 2003) updating task. Object-switch costs can be interpreted as the time cost for shifting the focus of attention to a new item within the set currently held in the direct-access region.

Similarly, on the procedural side, a to-be-performed target response is singled out by the response focus from the possible responses established by the task currently held in the bridge (e.g., red ink = left button press), and repeating the same response in the next trial is faster than changing to a different response (Bertelson, 1965). More recent research has shown that response-repetition benefits are obtained only in the case of a repetition of the task set as well. If the task switches, the effect is reversed, and response repetition produces costs (e.g., Druet & Hübner, 2008; Kleinsorge, 1999; Rogers & Monsell, 1995; Schuch & Koch, 2004). An analogous pattern in declarative WM has been demonstrated by research in our lab: Object-repetition benefits turn into object-repetition costs when the list switches concurrently (Oberauer, Souza, Druet, & Gade, 2011).

In sum, these findings show striking similarities in the selection of declarative and procedural representations, suggesting that declarative WM and procedural WM are structured analogously and operate according to similar principles. Despite these similarities, there have been only a few studies in which these selection processes were compared directly (for an exception see Risse & Oberauer, 2010). Therefore, the main goal of the present research was to evaluate whether the selection of declarative and procedural representations produces analogous behavioural signatures under equivalent experimental conditions.

Research questions and predictions

We aimed at evaluating two main assumptions derived from the model proposed by Oberauer (2009). The first hypothesis is that declarative WM and procedural WM are analogous structures that operate in similar ways to select, bind, and maintain representations available for processing. The second hypothesis is that these subsystems are independent of each other, such that people can hold in mind a set of declarative representations (e.g., a memory list) and a set of procedural representations (e.g., a task set) at the same time without mutual interference.

To evaluate these assumptions, we conducted two experiments in which we used sets of digits as the declarative representations and digit classification tasks as the procedural representations. College students worked through short runs of decisions to be made on digits held in memory. For each decision, two selections were required in both the declarative and the procedural WM system. On the declarative side, one of several memory sets and one digit within that memory set had to be selected as the object of the decision. On the procedural side, a task set and a response within that task set had to be chosen. The two higher level selections (i.e., memory set and task set selection) can be regarded as analogous processes in the two subsystems because they are assumed to exchange the whole content of the DA region in dWM and the bridge in pWM. These selection processes are reflected in memory-set switching

effects on the declarative side and task-set switching effects on the procedural side. Likewise, the two lower level selections (i.e., object and response selection) are analogous because they involve exchanging the contents of the attentional foci in the two subsystems. These lower level selections are reflected in object- and response-switch effects, respectively. In the present experiments, we focused on the comparison of memory-set switching and task-set switching to illuminate the higher level selection processes. The lower level selection processes are addressed in Oberauer et al. (2011).

At this point, we also want to highlight an important methodological innovation that we introduced into research on WM. In the WM research tradition, in which declarative contents received most attention, new memory sets are typically presented on every trial, thus asking participants to regularly establish new short-term bindings between memory contents and retrieval cues (e.g., bindings between list items and their list positions). This methodological practice for studying declarative WM contrasts with the one established for studying procedural WM. In the literature on choice response times, task switching, and multitasking, each task is defined by a constant set of stimulus–response mappings that is typically used throughout the experiment. Therefore, long-term associations between stimulus categories and responses (i.e., task sets) are established during the practice trials, before the experimental manipulations are implemented, and the task sets are assumed to be retrieved from LTM into WM whenever they are relevant (Mayr & Kliegl, 2000).

The use of new memory sets on every trial to study declarative WM is motivated by dual-store models, which distinguish a short-term store for brief maintenance of novel information from a more permanent long-term store. Many contemporary theories of WM have abandoned the distinction of WM and LTM in terms of separate stores and instead have conceptualized the former as a subset of the latter (Cowan, 1999; Postle, 2006; Ruchkin, Grafman, Cameron, & Berndt, 2003). The model of Oberauer (2009) stands in that tradition. From the perspective of these theories, it should not matter whether the memory sets are new on each trial or

well learned. Any set of memory contents that is selected and maintained available for ongoing processing is considered as the current content of WM. One consequence of this view is that empirical phenomena established with experiments using new memory sets on every trial should be replicable with memory sets that are established in LTM before the experiment begins. To provide a first partial test of this assumption and to methodologically equate the assessment of processes in dWM and pWM, we kept both the declarative and the procedural representations constant for each participant throughout the experiment. Hence, long-term associations between these representations and their retrieval cues could be formed.

Analogous effects in declarative and procedural WM

A central hypothesis within the framework proposed by Oberauer (2009) is that declarative and procedural subsystems have analogous structures, which operate by analogous principles. Therefore, one should expect that the effects commonly observed for declarative WM should reproduce for procedural WM, and vice versa. As discussed above, list-switch costs and task-switch costs are one example of analogous effects across these subsystems. Table 1 lists each component of the declarative and procedural WM and the set of analogous experimental effects that could be expected assuming that both subsystems operate analogously. For each of these hypothesized analogous effects it is also indicated whether it was confirmed (in previous studies or in the experiments reported in the present paper), disconfirmed, or remains to be tested. As indicated in Table 1, in the present experiments we aimed at reproducing on the declarative side a number of key findings that have been established for the procedural subsystem: switch costs, mixing costs, the effect of the number of sets, preparation effects, and residual switch costs.

Switch costs

Task-switch costs refer to the finding that switching from one task to another produces increased latencies and error rates when compared with task repetitions across trials. We expect to find

analogous list-switch costs. List-switch costs have been reported by Oberauer (2005), but that experiment differed in many ways from a typical task-switching experiment (e.g., each run consisted of only two trials). Therefore, it is important to demonstrate list-switch effects in an experiment that is more similar to the standard task-switch experiments.

Mixing costs

Mixing costs have been interpreted as reflecting task ambiguity: In mixed-task blocks, the context is ambiguous as to which task must be performed in the upcoming trial compared to single blocks, therefore producing a conflict in the selection of the appropriate task set. This conflict must be resolved on every trial, thus producing behavioural costs that are identifiable in the repetition trials (Rubin & Meiran, 2005). If one assumes that declarative representations are processed similarly to procedural ones, then mixing costs should also occur if the selection of declarative representations is to be accomplished in ambiguous conditions. Accordingly, we expect increased latencies and error rates in conditions in which people have to choose between more than one list compared to conditions in which a single list is to be used throughout.

Number of sets

When the number of task sets in mixed blocks is increased beyond two, mixing costs do not increase (Kray, Li, & Lindenberger, 2002; Rubin & Meiran, 2005). This finding supports the interpretation of mixing costs as reflecting the time for deciding which task has to be carried out: This decision must be made as soon as there is more than one candidate task set, regardless of how many there are. Consequently, we expect that list-mixing costs (in analogy to task-mixing costs) will not increase as the number of lists in a mixed block is increased beyond two.

With respect to switch costs, however, the pattern of previous results is less clear. Rubin and Meiran (2005) found that increases in the number of tasks from two to three did not influence task-switch costs; however, Kray et al. (2002) reported larger switch costs when participants had to switch

Table 1. Corresponding components of declarative and procedural WM and the analogous effects observed and predicted across the subsystems

<i>Declarative WM</i>	<i>Procedural WM</i>
Activated declarative LTM	Activated procedural LTM
1. List-mixing costs, not increasing beyond 2 lists (<i>present experiments</i>)	1. Task-mixing costs, not increasing beyond 2 tasks (Rubin & Meiran, 2005)
2. No set-size effect of currently irrelevant memory set on RTs (Oberauer, 2002, 2005)	2. No effect of number of S–R mappings of currently irrelevant task on RTs (predicted)
Region of direct access	Bridge
3. List-switch costs (Oberauer, 2005; <i>present experiments</i>)	3. Task-switch costs (Monsell, 2003)
4. No effect of number of lists on list-switch costs (disconfirmed by <i>present experiments</i>)	4. No effect of number of tasks on task-switch costs (Rubin & Meiran, 2005; <i>present experiment</i>)
5. Preparation time reduces list-switch costs, but leaves residual switch costs (<i>present Experiment 2</i>)	5. Preparation time reduces task-switch costs, but leaves residual switch costs (Rogers & Monsell, 1995)
6. N-2 list-repetition costs (predicted)	6. N-2 task repetition costs (Koch et al., 2010)
Focus of attention	Response focus
7. Object-repetition benefits in list-repetition trials and object-repetition costs in list-switch trials (Oberauer, Souza, Druey, & Gade, 2011).	7. Response-repetition benefits in task repetition trials and response-repetition costs in task-switch trials (Rogers & Monsell, 1995)
8. Object-switch costs increase with the set-size of list in DA region (Oberauer, 2002, 2003)	8. Response-switch costs increase with number of S–R mappings of the task set in the bridge (predicted)

Note: WM = working memory. LTM = long-term memory. RT = reaction time. DA = direct access. S–R = stimulus–response.

between four tasks than when they switched between two tasks. Given that, it seems difficult to predict whether increasing the number of tasks from two to three will increase task-switch costs in the present experiments. Nevertheless, whether task-switch costs increase or not as a function of the number of tasks, we expect to observe an analogous effect for list-switch costs.

One further conditional prediction can be made: If declarative WM and procedural WM operate independently, the number of tasks should not affect list-switch costs, and the number of lists should not affect task-switch costs.

Preparation for a set switch and residual switch costs

In task-switch experiments, *preparation* refers to the improvement in performance (i.e., faster responding and fewer errors) when participants know ahead of time which task they have to perform next. This is usually done by presenting a cue that indicates the task required in the next trial, followed by the imperative stimulus after a variable cue–stimulus interval (CSI). Provided that the CSI is long enough (i.e., 1,000 ms or more), task-switch

costs are usually reduced. Nonetheless, *residual switch costs* tend to persist even after very long preparation intervals and have been attributed either to response-related processes that cannot be completed before the stimulus is presented or to interference from other tasks (see Kiesel et al., 2010, for an overview). To evaluate whether preparation time also facilitates list switching, we varied CSI (300 vs. 2,000 ms) in Experiment 2, providing participants with ample time for switching to the new list in the long CSI. We expected that, as for the task-switch costs, list-switch costs would be reduced, albeit not eliminated, after a long CSI.

Selection of lists and tasks in WM

Our experimental task comprised the selection of a memory set and a task set in WM to perform a decision on digits, and repetitions and switches of the memory lists and tasks were manipulated independently. Therefore, this experimental task required selective access to two types of representation—declarative and procedural ones—and one might ask how these selection processes are

implemented in WM. We consider the differential predictions of three alternative views.

The first possibility is that selection of declarative and procedural representations shares a single processing bottleneck allowing just one step to be carried out at a time (Pashler, 1994). According to this view, a new list and a new task cannot be selected simultaneously because only one selection process can occupy the bottleneck at a time. Therefore, these operations have to run sequentially, and the time to run both of them is the sum of their individual durations, so that the time costs of switching between memory lists and task sets should be additive. The prediction of serial selection is depicted in the left panel of Figure 2.

An alternative possibility is to assume that these selection processes can be carried out independently and without mutual interference. This implies that list and task selection can run in parallel, and, therefore, the time to switch both the list and the task should be substantially lower than the time to switch each of these representations individually. According to this account, the declarative and procedural subsystems would not be constrained by the same processing bottleneck or limited resource (Risse & Oberauer, 2010). This hypothesis predicts an underadditive interaction of list switch and task switch, and this pattern is shown in the middle panel of Figure 2.

According to both hypotheses above, the selection of a new list and the selection of a new task

set are separate processing steps, as assumed in our theoretical framework (Oberauer, 2009). A radical alternative to this view could be formulated on the basis of the theory of event coding (Hommel, Müsseler, Aschersleben, & Prinz, 2001). According to this view, the stimuli and the task applied to them form a composite representation—an *event file*. This event file is the basic unit of selection. As a consequence, a repetition benefit can occur only if the same event file has to be selected on successive trials, but not if only one of its composite features (i.e., the stimulus or the operation) is repeated. Applied to our paradigm, this hypothesis would predict that a repetition benefit is observed only when both the task and the list repeat from one trial to the next. Partial repetitions (like when the task repeats but the list switches, or vice versa) would abolish any repetition benefit. Indeed, partial repetition may even produce a cost compared to full switches (Hommel, Proctor, & Vu, 2004). Hence, this hypothesis predicts a benefit for joint repetitions of lists and tasks, whereas partial repetitions and full switches would produce slower RTs, as illustrated in the right panel of Figure 2. This interaction differs from the one expected from parallel but separate switch processes: A hallmark of separate switch processes is that list-repetition benefits are still observed after a task switch, and task-repetition benefits are still observed after a list switch. This is the case in the predicted interaction pattern presented in the

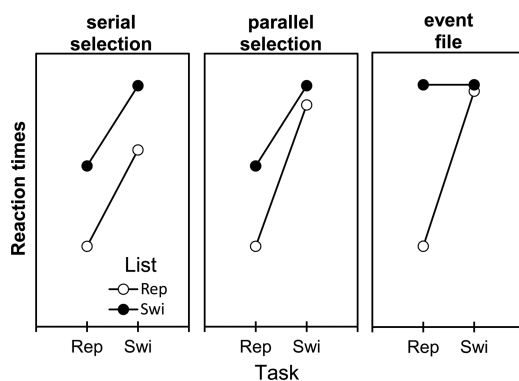


Figure 2. Schematic illustration of the possible interactions between list and task sequence (repetition, Rep; switch, Swi) according to three alternative hypotheses: serial selection (left panel), parallel selection (middle panel), and event file (right panel) hypotheses.

left and middle panels, but not in the right panel of Figure 2.

EXPERIMENT 1

In this experiment, participants learned three memory sets and three classification tasks. Each memory set comprised a list of three digits selected from the numbers 1, 2, 3, 4, 6, 7, 8, and 9. Each task set—a smaller–larger, an even–odd, and an outer–inner judgement—comprised a distinct cue (e.g., “iI” for the smaller–larger classification task), but all three tasks had the same response alternatives (to press the left or right arrow key). Memory sets and task sets were constant across the experiment.

Participants worked on runs of nine decisions on digits (hereafter called runs of trials). In each trial, they were prompted to retrieve one specific digit from one of the lists and to classify this digit according to one of the classification tasks. Within a run of trials, repetitions and switches of lists and of tasks were randomly determined, thereby allowing us to evaluate the time to switch just one or both of these representations. Furthermore, the set of possible relevant lists (one to three) from which participants had to select the digit and the set of possible task sets (one to three) from which the response had to be selected were varied orthogonally. Our manipulation of the number of lists and the number of tasks applies to runs: At the beginning of each run we specified one, two, or three memory sets and one, two, or three task sets as the candidates for the trials in that run. Across runs, all three lists and tasks (or the possible combinations of them) had an equal probability of being chosen as the candidate sets. Thus, for each run of trials, participants were previously informed about the set of candidate lists and the set of candidate tasks from which they had to select the appropriate ones for each decision. We assumed that once the candidate lists/tasks are determined, these would be activated in LTM, thereby becoming candidates for selection, whereas the remaining memory sets or task sets should be excluded from selection. We assumed that these relevant sets are the current

contents of WM. If this reasoning is correct, we should observe an effect of the number of candidate memory sets and of candidate task sets on decision latencies and error rates, even though all three memory sets and all three task sets are equally well established in LTM.

Method

Participants

Sixteen female students from the University of Zürich took part in return for either partial course credit or CHF 15 per session. The participants' age ranged from 20–34 years with an average age of 24 years.

Apparatus and stimuli

The experiment was administered on an IBM-compatible personal computer. The software was written in MatLab® 7, using the Psychophysics Toolbox extensions (Brainard, 1997; Pelli, 1997).

Participants performed three digit-classification tasks: a smaller–larger task, an even–odd task, and an outer–inner task. In the smaller–larger task, participants had to decide whether the target digit was smaller or larger than five. In the even–odd task, they were asked whether the target digit was even or odd. For the outer–inner task, participants had to decide whether the target digit was located on of the outer (1, 2, 8, and 9) or the inner (3, 4, 6, and 7) positions of a number line from one to nine. For all classification tasks, participants had to press the left arrow key on a standard keyboard (if the digit was smaller, outer, or even) or the right arrow key (if the digit was larger, inner, or odd) to give a response. Each task set comprised a task cue and an instruction. The instructions were the German words “kleiner–größer” (smaller–larger), “ausser–innen” (outer–inner), and “gerade–ungerade” (even–odd). The task cues were the symbols “iI” for the smaller–larger, “Oo” for the outer–inner”, and “-/” for the even–odd task. These task cues were designed to depict in an abstract fashion the nature of the task and the stimulus–response assignment (e.g., for the outer–inner task, the large O—representing the outer fringe

of the number series—was on the left, because the “outer” response was mapped to the left key).

Each to-be-encoded memory set (i.e., list) was displayed in a black rectangular frame on a white background with three black digits within each frame, positioned at the left, middle, and right (see Figure 3). Depending on the condition (which determined the number of relevant memory sets), one, two, or three such frames were displayed. The frames were arranged to appear in the top, middle, and bottom of the computer screen, centred horizontally. The vertical position of the frame was used to cue the respective list, and the horizontal position of a stimulus within the frame was used to cue the digit to be retrieved from that memory set. The digits comprised the numbers one to nine excluding five. Each list was constructed such that the three digits could never be all in the same category for any of the three categorization tasks (e.g., they were never all three odd, smaller than five, or on the inner positions of the number line), and all eight digits (1–4 and 6–9) were used at least once across the three lists. Because the composition of all three lists required nine digits, but only eight digits were available, the bottom list always included one digit that had already been used in the top or middle lists. Three lists were randomly generated with the above constraints for each participant and were held constant throughout the experiment.

Because the number of tasks and lists were manipulated across conditions, two displays, one showing the candidate task sets and one showing the candidate lists, were presented for self-paced encoding before each run of trials (explained in detail below). In the condition with one task set, the cue (e.g., “iI”) and the instruction (e.g., “smaller–larger”) were presented in the centre of the screen. In the condition with two task sets, cues and instructions for each task were presented in two lines, one above and one below the centre of the screen. Finally, in the condition with three task sets, one task set was presented in the top, one in the middle, and one in the bottom centre of the screen. Regarding the lists display, one, two, or three lists were presented at their respective top, middle, or bottom positions.

Procedure and design

The experiment was performed in three sessions, each consisting of two training blocks and three experimental blocks. The first session lasted one hour, and the following sessions lasted 30–40 minutes each. At the beginning of the first session, participants received written instructions regarding the experimental task, and these instructions were also available for consultation in the next sessions.

In the first training block, participants had to encode and accurately retrieve the three lists of three digits. Each trial consisted of self-paced encoding of all three lists, followed by the recall of all list items. In the recall part, all three list frames were presented, and a question mark was displayed at a random digit position within one of the frames. All nine digits were thus probed for recall in random order. Participants had to enter the correct digit using the alphanumeric keyboard. Training continued until all lists were accurately recalled three times in succession (first session) or once (second and third sessions).

The second training block comprised the practice of both lists and task sets. In this block, a run of trials started with the presentation of the three task sets simultaneously on the first screen. When participants pressed the space bar, one list of digits selected at random from the three lists was displayed in its respective top, middle, or bottom position. When the participant pressed the space bar again, a series of nine trials commenced. In each trial, the empty list frame was presented, and one task cue (“iI”, “Oo”, or “-/-”) was displayed at a random digit position within that frame until a response was made. The participant was instructed to retrieve the digit in the list position marked by the location of the task cue and to apply to that digit the categorization indicated by the task-cue symbol, entering the response by pressing the appropriate key as quickly as possible. This training block continued until all responses were accurate on three consecutive runs of nine trials (first session) or on one run (second and third sessions).

After the training blocks, the actual test blocks of the experiment started. Participants completed three test blocks per session. Each block consisted

of 24 runs of trials, and each run comprised the following sequence of events (see Figure 3). First, a display with the candidate task sets (one, two, or three) was presented, followed by a display with the candidate lists (one, two, or three)—both for self-paced encoding. The number of tasks and lists in each trial varied depending upon the experimental condition (see below). Then a series of nine trials followed, in which the empty list frames were displayed, and a task cue was shown at a random digit position in one of the frames until a response was made. As in the training block, the position of the task cue indicated the list and the digit to be retrieved from that list (i.e., the frame and position within the frame, respectively), and the task-cue symbol indicated the task to be performed on the digit.

From one trial to the next, repetitions and switches of the list and task were determined at random and independently of each other. The probability of a repetition (one, one half, and one third) varied according to the number of lists and tasks (one, two, or three, respectively) in that condition. Whenever a list was repeated, the same digit as that in the previous trial could be selected as the target of an operation with a one-third probability. In list-switch trials, however, the selection of the same digit as that in the previous trial was a rare event (less than 4% of the trials); therefore, we considered all of these trials as a case of the selection of a new digit.

In all blocks, verbal feedback (the German words “Richtig” or “Falsch!”) was provided for correct and incorrect responses, respectively, for 500 ms, followed by a new trial after 100 ms. After nine trials had been completed, a new run began with the selection of new candidate lists and tasks. In between blocks, participants were allowed to take a short break.

Each testing block comprised a different combination of number of tasks (NTasks) and number of lists (NLists). Nine different conditions were programmed—1:1, 1:2, 1:3, 2:1, 2:2, 2:3; 3:1, 3:2, and 3:3, with the first digit standing for the number of tasks and the second digit for the number of lists. For example, in the 1:1 condition, each run comprised the presentation of one

randomly selected task set and one randomly selected memory set, which was then held constant for the subsequent nine trials. The three tasks and three lists were equally likely to be selected in each run of trials such that, across runs, all lists and all tasks were performed, but within one run, the same task and list were repeatedly used. In the 2:2 condition, each run involved two task sets and two memory sets. The pairs of tasks were randomly selected from the possible available combinations, (iI) and (Oo), (Oo) and (-\), or (iI) and (-\). The two memory lists were likewise selected from the three possible combinations: top and middle, middle and bottom, or top and bottom lists. Again, across runs, all lists and tasks were equally likely to be used, but within one run, only the selected lists and tasks were used. In condition 3:3, each run involved all three task sets and all three memory lists.

Half of the participants were given the following order of conditions—3:1, 2:2, and 1:3 in the first session; 2:3, 1:1, and 3:2 in the second session; and, 1:2, 2:1, and 3:3 in the third session. The other half received the reverse order of conditions.

Classification of switch conditions and data trimming

Reaction times (RTs) for incorrect trials (5.3%) and trials following incorrect responses were removed (10.5% of the data altogether). Additionally, RTs faster than 300 ms and longer than 6,000 ms were excluded (0.4%). Mean reaction times (RTs) were computed for each of four different switch categories: (a) list and task repetition; (b) list repetition and task switch; (c) list switch and task repetition; and (d) list and task switch. In all these switch categories, only trials that required the selection of a new digit were considered to avoid confounds with digit-repetition effects. The first decision in each trial could not be classified in any of those categories and was excluded (11.1%).

The experimental conditions differed with respect to the number of possible switch categories. In conditions with only one task set (i.e., conditions 1:1, 1:2, and 1:3), only task repetitions were possible, and in conditions with only one list (i.e.,

conditions 1:1, 2:1, and 3:1), only list repetitions were possible. Thus, the full set of switch conditions was available only from the conditions 2:2, 2:3, 3:2, and 3:3.

Results

Repeated measures analyses of variance (ANOVAs) were conducted to assess the effects of the independent variables (i.e., list switch, task switch, number of lists, and number of tasks) on RTs and percentage of errors (PE). For some analyses, the sphericity assumption was violated, and hence corrected Greenhouse–Geisser degrees of freedom (recognizable by noninteger values) were reported. The statistical analyses were performed using log-transformed RTs because their distributions better approximate normality; the pattern of significance was similar for nontransformed RTs. In addition, partial eta-squared (η_p^2) values are reported as measures of effect size for the significant results.

Figure 4 presents RTs as a function of repetitions or switches of lists and of tasks, with separate panels for each combination of number of lists and number of tasks. RTs were faster when the same list and task were repeatedly accessed, but increased as a function of switching demands, with switches of both lists and tasks producing the slowest RTs.

The PE was plotted in the same way in Figure 5. Error rates were also affected by the experimental manipulations. Error rates increased with increases in the number of lists and tasks and as a function of the switching requirements.

We analysed specific contrasts to test the hypotheses outlined in the introduction. Report of the results is ordered by these predictions.

Switch costs

We first analysed the effects of list-switching and task-switching effects separately. To assess the effect of list switching, we carried out an ANOVA with list-switch condition, number of lists, and number of tasks. Only task-repetition trials were included to avoid distortion of the list-switch effect by its interaction with task switching

(to be discussed below). Analogously, the effect of task switching was assessed with an ANOVA with task-switch condition, number of lists, and number of tasks, including only list-repetition trials. All conditions were included in these analyses, with the exception of conditions with just one list (i.e., conditions 1:1, 2:1, and 3:1 in the computation of list-switch costs) and conditions with just one task (i.e., conditions 1:1, 1:2, and 1:3 in the computation of task-switch costs).

List-switch costs. Accessing a new digit within the same list was faster than accessing a new digit in a different list, $F(1, 15) = 51.17$, $p < .001$, $\eta_p^2 = .77$, as can be seen for each individual condition in Figure 4. The corresponding analysis of PE did not show a significant effect of list switching (mean values of 5.0% and 5.3%, $F < 1$; see Figure 5). List-switch costs in RTs (i.e., the difference between list-switch and list-repetition trials) increased as the number of lists increased from 2 to 3 (see Figure 6, top panels), as reflected in the List Switching \times NLists interaction, $F(1, 15) = 8.40$, $p = .011$, $\eta_p^2 = .36$. Again, the corresponding analysis on PE was not significant ($F < 1.5$). List-switch costs did not increase with the number of tasks ($F < 1$, for both the RTs and PE). This pattern is as expected: Switching to a new list became more difficult as the number of lists to choose from increased, but was independent of the number of task sets.

Task-switch costs. Similarly to list switching, switching the task slowed responding compared to repeating the same task, $F(1, 15) = 71.79$, $p < .001$, $\eta_p^2 = .83$ (Figure 6, bottom panels). These costs were not affected by the number of tasks ($F < 1$ for the relevant interaction), but they increased with the number of lists, as reflected in the significant Task Switching \times NLists interaction, $F(2, 30) = 28.65$, $p < .001$, $\eta_p^2 = .66$. Repeated contrasts indicated that task-switch costs did not increase when the number of lists was increased from one to two: for the 2-tasks condition, $F(1, 15) = 2.1$, $p = .17$, $\eta_p^2 = .12$, and for the 3-tasks condition, $F(1, 15) = 1.1$, $p = .31$, $\eta_p^2 = .07$. However, task-switch costs were higher in the 3-lists condition

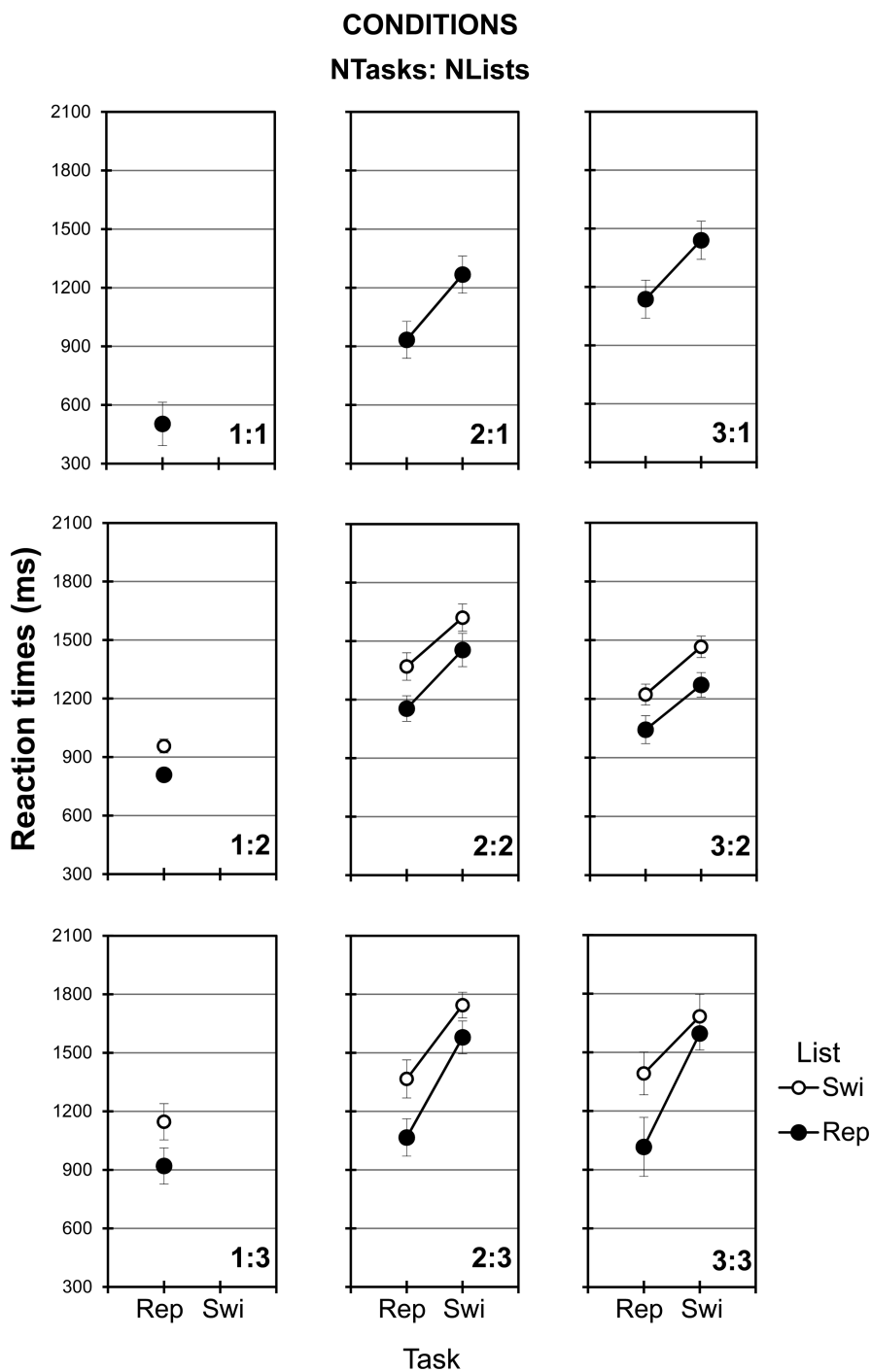


Figure 4. Reaction times as a function of repetition (Rep) and switches (Swi) of the task and list in each condition of Experiment 1. The number of tasks (NTasks) and the number of lists (NLists) in each condition are depicted inside each panel. Error bars represent 95% confidence intervals for repeated measures.

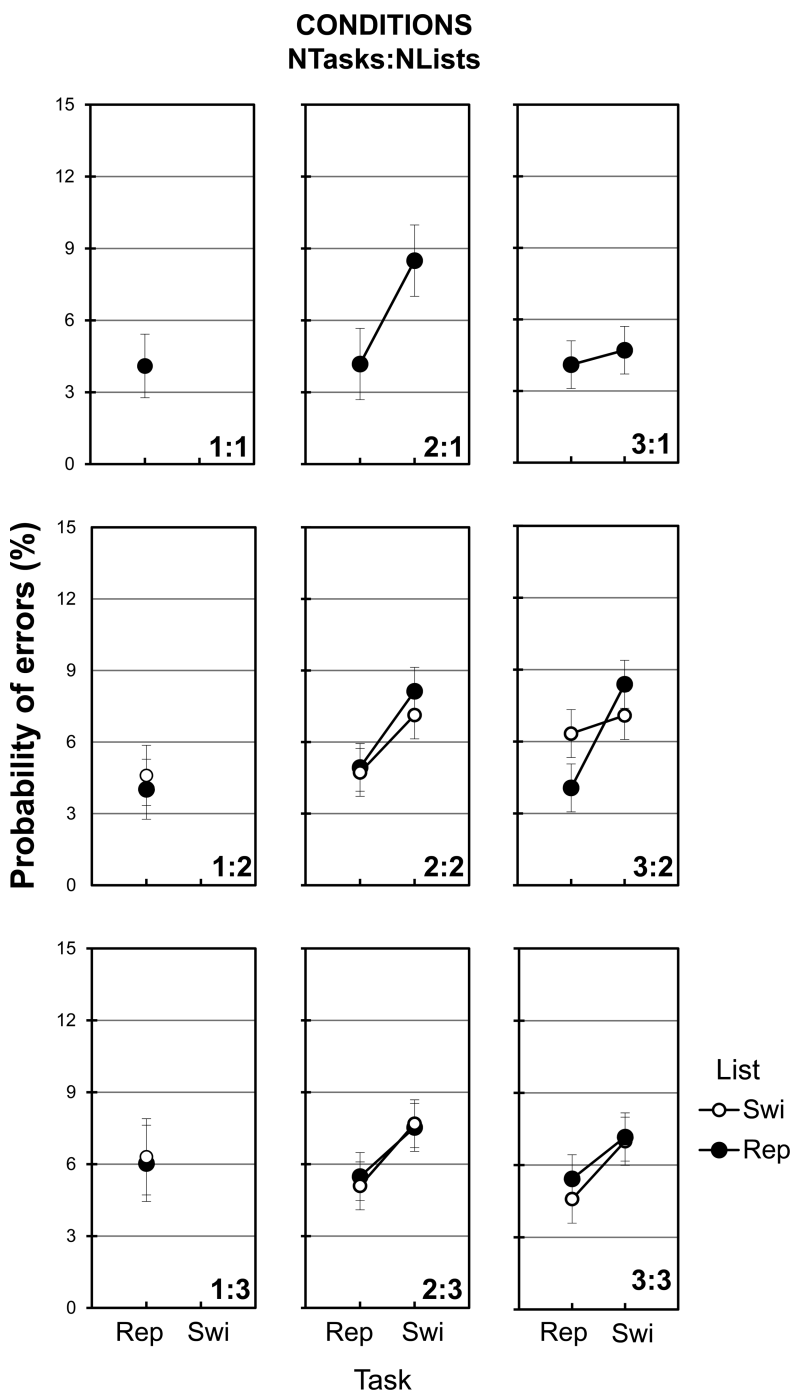


Figure 5. Percentage of errors as function of repetition (*Rep*) and switches (*Swi*) of the task and list in each condition of Experiment 1. The number of tasks (*NTasks*) and the number of lists (*NLists*) in each condition are depicted inside each panel. Error bars depict 95% confidence intervals for repeated measures.

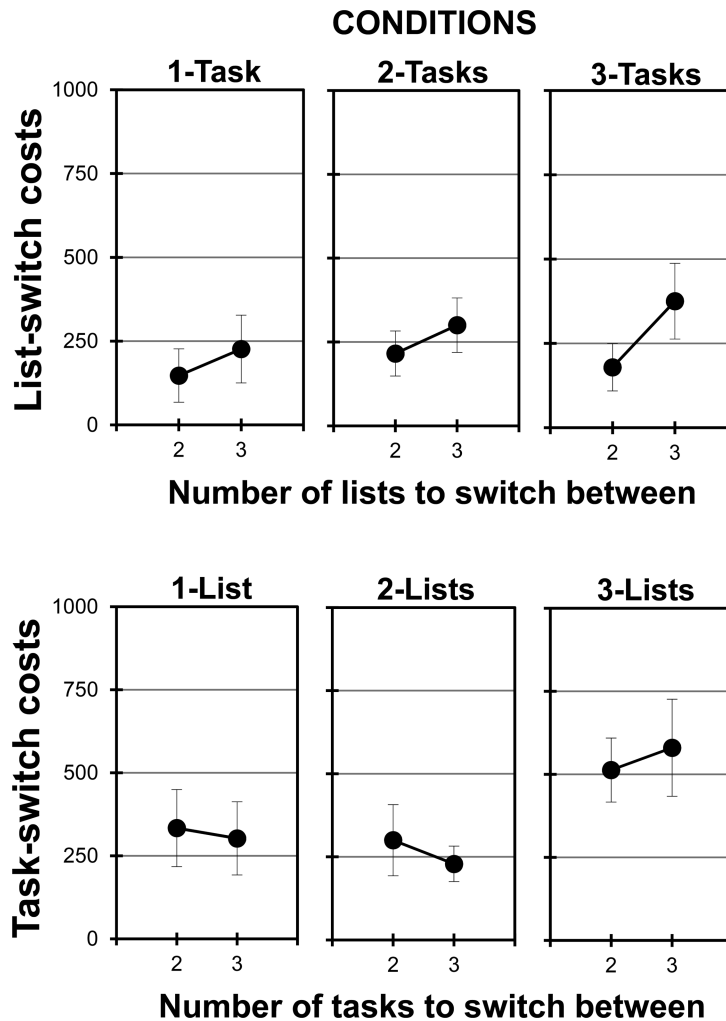


Figure 6. List-switch costs and task-switch costs as a function of the number of lists and tasks in Experiment 1. Error bars show 95% within-subjects confidence intervals for repeated measures.

than in the combined 1-list and 2-lists conditions: for the 2-tasks condition, $F(1, 15) = 11.0$, $p = .005$, $\eta_p^2 = .42$; and for the 3-tasks condition, $F(1, 15) = 33.1$, $p < .001$, $\eta_p^2 = .69$. Task switching also incurred a cost in terms of errors (7.4 vs. 4.7%), $F(1, 15) = 11.43$, $p = .004$, $\eta_p^2 = .43$. However, none of the interactions involving task switch was significant in the PE ($F < 1$).

In sum, the pattern of interactions goes against our expectations: The cost of selecting a new task did not depend on the number of

tasks to select from; however, it did depend on the number of lists maintained available in declarative WM.

Mixing costs and number of sets

Mixing costs are well established in procedural WM (Rogers & Monsell, 1995): RTs on no-switch trials are increased in mixed blocks with more than one task, compared to pure blocks in which only a single task is to be carried out. Here we investigated whether the pattern of mixing

costs is similar for memory sets and task sets. We evaluated mixing costs by comparing RTs during list and task repetition trials across all experimental conditions using a 3 (NLists) \times 3 (NTasks) ANOVA. RTs increased with a larger number of lists, $F(2, 30) = 29.98$, $p < .001$, $\eta_p^2 = .67$, and with a larger number of tasks, $F(2, 30) = 88.49$, $p < .001$, $\eta_p^2 = .86$. Thus, there were mixing costs both in procedural WM and in declarative WM.

The effects of number of lists and of number of tasks showed the same pattern, increasing from 1 to 2 and then levelling off. The mean RT for the conditions with one list was 857.9 ms (95% confidence interval, CI, for within-subject comparisons = 51.5), with two lists was 1,002.2 ms (44.4), and with three lists was 1,000.9 ms (59.8). The mean RT for the conditions with one task was 744.2 ms (65.6), with two tasks was 1,051.2 ms (44.2), and with three tasks was 1,065.6 (41.2). The discontinuous trends were confirmed by repeated contrasts. They indicated that RTs increased significantly when the number of sets increased from one to two: $F(1, 15) = 51.22$, $p < .001$, $\eta_p^2 = .77$, for the lists, and $F(1, 15) = 98.25$, $p < .001$, $\eta_p^2 = .87$, for the tasks. There was, however, no significant difference between two and three sets, and this holds for both memory sets and task sets ($F < 1$, $p = .42$, $\eta_p^2 = .04$, for the lists, and $F < 1$, $p = .55$, $\eta_p^2 = .02$, for the tasks). In addition, the ANOVA resulted in a significant NLists \times NTasks interaction, $F(1.5, 22.3) = 8.91$, $p = .003$, $\eta_p^2 = .37$, because in the condition with three tasks the effect of the number of lists was not significant. No significant effects were obtained for the error rates ($F < 2$).

Parallel selection of lists and tasks

In conditions 2:2, 2:3, 3:2, and 3:3, all types of combinations between list repetitions and switches and task repetitions and switches were possible, thus allowing the evaluation of list and task repetition benefits and the interaction between list and task switching. As can be seen in Figure 4, repeating the same list produced list-repetition benefits both in task-repetition, $F(1, 15) = 45.56$, $p < .001$, $\eta_p^2 = .75$, and in task-switch trials, $F(1, 15) = 51.87$, $p < .001$, $\eta_p^2 = .78$. Likewise, task-

repetition benefits were observed both in list-repetition, $F(1, 15) = 75.3$, $p < .001$, $\eta_p^2 = .83$, and in list-switch trials, $F(1, 15) = 50.55$, $p < .001$, $\eta_p^2 = .77$. This pattern suggests that lists and tasks were not selected as a unified compound, but separately, such that a list remains selected when the task is switched, and a task remains selected when the list is switched. Furthermore, list-switch and task-switch costs interacted underadditively, $F(1, 15) = 17.72$, $p = .001$, $\eta_p^2 = .54$. Therefore, the pattern of results is most in line with the predictions of two selection processes running in parallel (see Figure 2, middle panel).

Discussion

In the present experiment, the selection of declarative and procedural representations in working memory was assessed, and analogous effects were observed in most comparisons. First, we observed that repeatedly accessing the same list was 147 ms (1:2 condition) to 375 ms (3:3 condition) faster than switching to a different list, and the same was true for task repetitions and switches (with time costs ranging from 228 ms in condition 3:2 to 580 ms in condition 3:3). Time costs associated with switches between two memory sets were first reported by Oberauer (2005). However, that study differed in important regards from the typical task-switching paradigm, in that the memory lists were new on every trial, and participants performed just two successive decisions on the items from each pair of memory lists, with the second decision comprising either a list repetition or a list switch. Therefore, our results show for the first time that the requirement to randomly access a well-learned but currently not selected memory list produces substantial list-switch costs, comparable to the costs associated with switching between task sets.

Second, similar patterns of mixing costs were obtained for lists and tasks. We replicated mixing costs in RTs for task sets in procedural WM and extended this finding by analogy to declarative WM where we obtained mixing costs for memory sets. The discontinuous pattern of mixing costs in procedural working memory, with an increase in RTs from one to two task sets but no further

increase with larger numbers of tasks, replicates previous findings (Hübner, Futterer, & Steinhauser, 2001; Kray et al., 2002; Rubin & Meiran, 2005). We found the same discontinuous pattern for mixing costs in declarative working memory, further supporting our contention that the two forms of mixing costs reflect analogous processes in declarative and procedural working memory.

The analogy between declarative and procedural WM met its limits in the effects of the number of lists and the number of tasks on list-switch costs and task-switch costs: Whereas the list-switch costs were affected by the number of lists to be switched among, no analogous effect of the number of tasks on task-switch costs was found. This latter finding is in agreement with Rubin and Meiran (2005), but not with Kray et al. (2002), who found task-switch costs to increase with the number of task sets. Because of the finding of Kray et al., the possibility remains that there is an effect of number of tasks on task-switch costs, which we, as well as Rubin and Meiran, just failed to detect. If there is actually no effect of number of tasks on task-switch costs, this would point to a difference between selection of lists in declarative WM and selection of tasks in procedural WM.

A possible explanation for this difference could be that, when a new task set is selected, the old task set is rapidly removed from the bridge and is replaced by the new set. Therefore, there is never more than one task set in the bridge at the same time (see Mayr & Kliegl, 2000). This suggestion is in accordance with the observation that preparation for a task switch tends to be fully completed within a short time (~ 600 ms; e.g., Meiran, 2000; Monsell, Sumner, & Waters, 2003; Rogers & Monsell, 1995). In contrast, the DA region can potentially accommodate more than one list at the same time. Therefore, when a new list is selected, it is retrieved from activated LTM into the DA region, while the old list is still not completely removed from the DA region. This idea is consistent with the finding that when one of two memory sets is declared (temporarily) irrelevant, its removal from the DA region takes about 1 to 2 seconds. During that time, set-size effects of

both lists are observed, implying that both lists are still in the DA region (Oberauer, 2001, 2002, 2005). When participants in the present experiment switched between two lists, remnants of one currently irrelevant list might have remained in the DA region. When they switched between three lists, remnants of two currently irrelevant lists would be lingering in the DA region. In sum, because removal of no-longer-relevant information from the DA region is sluggish in comparison with the rapid removal of task sets from the bridge, a larger number of lists used in a run results in a larger load on the DA region.

The question arising from this hypothesis is: Why should the capacity limit of the bridge be strictly constrained to only one set, whereas the DA region would be able to accommodate more than one set at the same time? One possibility is that the degree of interference among representations is different in declarative and procedural WM. Mayr and Kliegl (2000) suggested that task sets strongly compete with each other for selection. To avoid the selection of the wrong task set (and therefore to shield the system from the repeated execution of strong habits), a mechanism is required to protect the currently relevant task set from the interference from previously used, but currently irrelevant, sets. In line with this possibility, it has been demonstrated that the abandoned task set is inhibited in order to reduce its likelihood to interfere with the implementation of the new task (Mayr & Keele, 2000; see Koch, Gade, Schuch, & Philipp, 2010, for a review). Thus, it seems that the strong constraint in the bridge might be related to the need to protect it from the competition among tasks. Conceivably, interference between two lists in declarative WM is less severe than interference between two tasks in procedural WM. One reason for that could be that there is less overlap between two memory lists than there is between two task sets. For instance, in our experiments, the memory lists shared at best one digit, whereas all three task sets applied to the same set of objects (i.e., digits) and assigned them to the same two responses.

Another unexpected result was the effect of the number of lists upon task-switch costs. Does this

result suggest that declarative and procedural WM share a single capacity? This is unlikely: The assumption of a single capacity for both subsystems makes two predictions that were disconfirmed by our data: First, the load effect would be a monotonically increasing function of the number of lists, such that increases from one to two lists should also have reduced the available capacity. Contrary to that prediction, task-switch costs did not increase from the 1-list to the 2-lists condition; they only increased between 2 and 3 lists. Second, a shared capacity predicts symmetric trade-offs between the two subsystems: The number of tasks should increase the load on the common capacity in the same way as the number of lists. Again, this prediction was not confirmed: We found no effect of the number of tasks on task-switch or list-switch costs. Thus, on balance, the evidence does not favour a shared capacity across the two subsystems.

We can only offer a speculative explanation for the unpredicted effect of number of lists on task-switch and list-switch costs. With three lists to keep in the DA region, participants might have tried to aid maintenance by articulatory rehearsal of the lists. Several studies have shown that overt articulation of irrelevant material (so-called articulatory suppression) can increase task-switch costs (Baddeley, Chincotta, & Adlam, 2001; Miyake, Emerson, Padilla, & Ahn, 2004). Arguably, subvocal articulation of digit lists could have the same effect. Therefore, the effect of number of lists on task-switch costs could have emerged from participants' subvocal articulation of the digit lists in the condition where all three lists were used.

It is worth noting in this context that Liefoghe, Barrouillet, Vandierendock, and Camos (2008) also obtained asymmetric interference between declarative memory load and a concurrent task-switching demand, but in the opposite direction from ours: Whereas increasing declarative load had no effect on task-switching costs, recall from declarative WM was impaired when a concurrent series of digit classifications involved many task switches. Further work with the same paradigm by Barrouillet, Portrat, and Camos (2011) also demonstrated an adverse effect of concurrent

processing tasks on performance of declarative WM (i.e., serial recall of letter lists). At first glance, these results could be interpreted as suggesting a shared capacity of declarative and procedural WM. However, these experiments also illuminate why concurrent processing impairs declarative WM: These experiments all allow a fixed limited time for the concurrent processing task, and the proportion of available time that is actually required for the processing task determines how much performance on the declarative memory task is impaired. Once this temporal variable is held constant, it does not matter for declarative WM performance how demanding the concurrent processing task is. Our interpretation of these findings is that the concurrent processing task impairs declarative WM because it introduces interfering material (i.e., the stimuli to be processed). The more of the available time is used up by the processing task itself, the less is left for restoring representations in declarative WM after interference. Thus, the concurrent processing task interferes with declarative WM indirectly, by introducing interfering declarative representations and by stealing time for compensatory measures, but the results of Liefoghe et al. (2008) and Barrouillet et al. (2011) do not suggest a shared capacity for maintaining declarative and procedural representations available simultaneously.

Finally, our findings also speak to the independence of selection of representations in declarative and procedural WM. We found substantial list-repetition benefits in task-switch trials, and likewise task-repetition benefits were observed in list-switch trials (Figure 4, conditions 2:2, 2:3, 3:2, and 3:3). These repetition benefits indicate that one can change the declarative representations while keeping the procedural ones selected, and vice versa. Given that declarative and procedural representations are selected separately, their selection can either proceed in parallel, which should produce underadditive effects of list switching and task switching, or compete for a single processing bottleneck and therefore be carried out serially, which should produce additive effects of list switching and task switching (see Figure 2). Our results indicated that switching both the list and the task

at the same time produced smaller costs than the sum of the individual switch costs, thus revealing an underadditive interaction, as predicted by separate and parallel selection of declarative and procedural representations (cf. Risse & Oberauer, 2010).

EXPERIMENT 2

Experiment 2 introduced a variation of preparation time between a list-switch cue and the stimulus indicating which digit is to be classified by which task. In the case of list switches, the preparation time can be used to remove the previous list from the DA region and to retrieve the new list from activated LTM and establish it in the DA region. One purpose of the manipulation of preparation time is to test our ad hoc explanation of why the number of lists affected list-switch costs (and task-switch costs). If the number-of-list effect occurred because, after a list-switch, the previous list was only incompletely removed from the DA region, then allowing longer preparation time than that in Experiment 1 should eliminate the effect of number of lists on list-switching costs. The second purpose of Experiment 2 was to investigate whether the effect of preparation time on list switching is analogous to the effect of preparation time on task switching.

Preparation effects have been extensively investigated in the context of task switching (Kiesel et al., 2010). It has been demonstrated that when participants know ahead of time which task is going to be relevant on the next trial, they can configure the cognitive system for the upcoming task, thus reducing the costs associated with task switches. However, it has also been established that complete advance task preparation is not possible: Even after long preparation intervals (more than 1,000ms between task cue and stimulus), residual switch costs remain.

Participants learned three lists of three digits each. The interval between the presentation of the list cue and the cue specifying the target digit within that list was varied. This interval enabled preparation of the new list, but it could not yet be

used to carry out the required decision because the target digit was not yet specified. We also varied the number of lists that participants had to access within a run of trials (one, two, or three). The task in each run of trials was to classify the cued digit according to just one classification task (a smaller–larger, even–odd, or outer–inner judgement). These manipulations allowed the replication of the main list-switching effects observed in Experiment 1 (i.e., list-switch costs and list-mixing costs) and to evaluate how these costs are affected by preparation.

Method

Participants

Seventeen female students from the University of Zürich took part in return for either partial course credit or CHF 15 per session. The participant's age ranged from 19 to 31 years with an average age of 23 years. None of the participants had served in the previous experiment.

Apparatus and stimuli

The apparatus and programming software were the same as those in Experiment 1. The memory lists to be used on each run (one, two, or three lists) were displayed by presenting a centred row with three coloured frames on a white background. The lists were composed of digits from one to nine, excluding five. Within each frame, a digit was presented in black colour as illustrated in the left panel of Figure 7. The colour of the frame was used to cue the respective list (red, blue, or green). The position of the frame (i.e., left, middle, or right) served to cue the digit to be retrieved from that list. Each list set was constructed such that the three digits could never belong to the same classification category, and that each list repeated one digit from each of the other lists. Repeated digits always appeared in different positions across lists. Therefore, six different digits were used to create the three memory lists, and three of these digits were repeated across the lists. For example, six digits were selected at random (1, 3, 4, 6, 7, 8), and the three lists were created as follows: (8, 1, 4), (3, 6, 1), (7, 8, 3). As can be seen in this

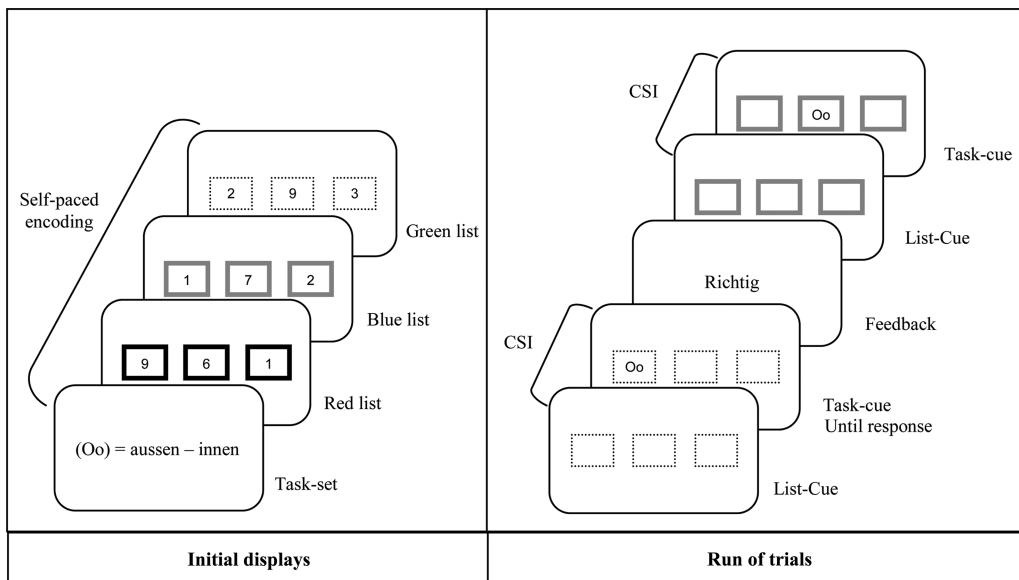


Figure 7. Example of the initial displays (left panel) and of the run of trials (right panel) used in Experiment 2.

example, all lists comprise digits drawn from all possible categories (small and large, odd and even, outer and inner). In addition, three of the digits are repeated across two of the three lists (1, 3, 8), and three digits are not repeated (4, 6, 7). Each list remained on the screen until the spacebar was pressed. The order of presentation of the three lists for encoding was fixed (red, blue, and green). For each participant, a computer program randomly constructed the list sets, which were maintained constant throughout the experiment.

The same classification tasks (smaller–larger, outer–inner, and even–odd) with the same response assignments as those described in Experiment 1 were used, but participants never had to switch between tasks. Participants again performed three sessions, each with a different task. Task order was randomly determined for each participant.

Procedure and design

Each session consisted of two training blocks and three experimental blocks. The first session lasted about 1 hour, and the following sessions lasted 40–50 minutes. At the beginning of each session, participants received written instructions regarding the experimental task.

In the first training block, participants had to encode and accurately retrieve the three lists of digits. List training was the same as that described for Experiment 1. The second training block comprised the practice of the lists and the classification task assigned to the session. In this block, a trial comprised: (a) the presentation of the assigned task, (b) the presentation of the list sets—both for self-paced encoding, and (c) a series of 13 decisions (the run of trials). In each trial, the row of frames was displayed (in red, blue, or green), and then the cue for the assigned task (“il”, “Oo”, or “-”) was presented at a random digit position until a left–right response was made. The list and the target digit were randomly selected in each trial; the task remained the same throughout the run. This training continued until responses were accurate on nine runs (first session) or on three runs (second and third sessions) in succession.

Next, participants completed three test blocks, each comprising a different number of lists (one, two, or three); the order of these three conditions was determined randomly for each session and participant. Additionally, each block was divided into two subblocks with equal numbers of trials, but different CSIs. The order of the CSI subblocks

within a block was also randomly determined. In the 1-list condition, participants had to hold one list available for processing during the run of trials. A new relevant list was selected at random for each run with equal probability for each of the three pretrained lists. In the 2-lists condition, two of the three lists were randomly selected to be used in the current run, and all possible combinations had equal probability for being selected between runs. Finally, in the 3-lists condition, digits from all three lists were eligible as the target of a decision operation. For all conditions, only the selected lists for the upcoming decisions were presented for self-paced encoding, and then decisions upon the digits from these lists followed. List repetitions and switches were determined randomly with the probabilities of a list repetition (one, one half, and one third) depending on the number of candidate lists in the run (one, two, and three, respectively). The probability of a digit repetition was one third for list repetitions and one sixth for list switches. Participants performed 10 runs (i.e., a total of 130 trials) in the 1-list condition, and 20 runs (260 trials) in each of the remaining conditions (i.e., 2- and 3-lists conditions) per session.

In all trials, the coloured frames (i.e., the list cue) were presented first. After a CSI (either 300 ms or 2,000 ms), the task cue was displayed in one of the list frames until responding. Feedback was provided after each response for 500 ms (the German words "Richtig" and "Falsch!"), which in turn was followed by the next trial after a blank 500-ms interval.

Classification of switch condition and data trimming
RTs for incorrect responses and following incorrect responses were excluded (4.4%), as well as RTs longer than 6,000 ms and faster than 250 ms (0.4%). Additionally, trials that included digit repetition were removed from the analysis (36%). The remaining trials were separated into list-repetition and list-switch trials. Mean RTs and error rates for these switch categories were computed separately for the short (300 ms) and long (2,000 ms) CSI values.

Results

Repeated measures ANOVAs were conducted to evaluate list-mixing costs, list-switch costs, and list-preparation effects. As for the previous experiment, log-transformed RTs were analysed. The pattern of significant results was the same with nontransformed RTs, unless explicitly noted. The data from all three sessions were collapsed, because in a preliminary analysis the choice of classification task did not affect the main results.

List-mixing costs

Our analysis of mixing costs focused on list-repetition trials (which were equivalent to the list- and task-repetition trials in the previous experiment). The left panel of Figure 8 presents the mean RTs and PE in the list-repetition trials as a function of the number of lists and CSI. The main effect of the number of lists on RTs was significant, $F(2, 32) = 20.75, p < .001, \eta_p^2 = .57$. Repeated contrasts indicated that moving from one to two lists produced mixing costs, $F(1, 16) = 43.97, p < .001, \eta_p^2 = .73$, but adding a third list did not increment these costs, $F < 1, p = .49, \eta_p^2 = .03$. The main effect of CSI was also significant, $F(1, 16) = 7.53, p = .014, \eta_p^2 = .32$, because the shorter CSI produced faster RTs than the longer one. List-mixing costs were reduced by preparation as reflected in the significant $NLists \times CSI$ interaction, $F(2, 32) = 17.09, p < .001, \eta_p^2 = .52$. No significant effects were obtained in the analyses of PE.

List-switch costs and list preparation

We examined list-switch costs by comparing list-repetition trials with list-switch trials in a $2 (CSI) \times 2 (switching) \times 2 (number\ of\ lists)$ repeated measures ANOVA. Figure 8 also shows the list-switch effects for the 2-lists and 3-lists conditions.

Switching to a different list produced reliable time costs, $F(1, 16) = 93.97, p < .001, \eta_p^2 = .86$, which was modulated by an interaction with CSI, $F(1, 16) = 38.90, p < .001, \eta_p^2 = .71$, thus indicating that list-switch costs were reduced, albeit not eliminated, by the long CSI. Moreover, the three-way interaction was marginally significant, $F(1, 16) = 3.36, p = .085, \eta_p^2 = .17$, because list-switch

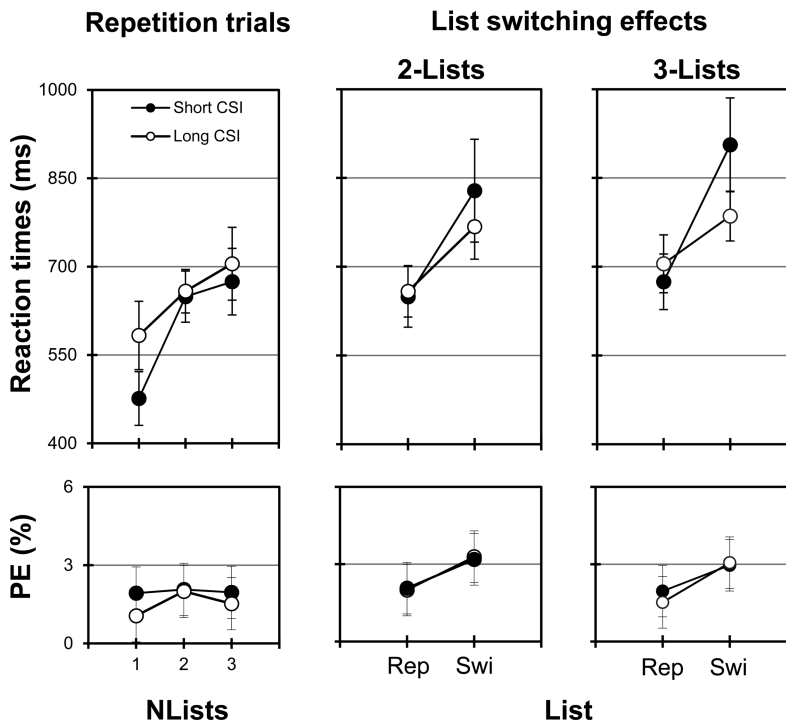


Figure 8. Mean reaction times (RTs; top panels) and percentage of errors (PE; bottom panels) in the analysis of list-mix costs (list-repetition trials) and list-switching effects (list-repetition vs. list-switch trials) for each value of the cue-stimulus interval (CSI; short vs. long,) in Experiment 2. Error bars represent 95% confidence intervals for repeated measures.

costs tended to be higher under the 3-lists condition than under the 2-lists condition for the short CSI, but not for the long CSI. Regarding the PE, the only significant effect was of list switching, $F(1, 16) = 9.33$, $p = .008$, $\eta_p^2 = .37$, indicating that list switching increased the error rates compared to list repetition.

Discussion

In the present experiment, we aimed at further evaluating analogous effects across declarative and procedural WM. More specifically, we asked whether list mixing, list switching, and list preparation would produce analogous effects to those reported in the task-switching literature. The results mostly confirmed analogous processing principles.

Whereas in Experiment 1, position cues were used for both the list and its items, and both were cued at

the same time, in Experiment 2, we used different cues for the list and the items (i.e., colours and the position of the frame, respectively), which were presented temporally separated. In this situation, we still observed fairly large list-switch costs, ranging from 179 ms to 231 ms for the 2-lists and 3-lists conditions, respectively, under the short CSI. The opportunity to prepare for a list switch during the longer CSI (2,000 ms) significantly reduced switch costs. Yet, even after a long CSI, we still observed residual switch costs of 109 ms and 81 ms for the 2-lists and 3-lists conditions, respectively.

In addition, we replicated the discontinuous pattern of list-mixing costs, first seen in Experiment 1, in an experiment without the added need to switch tasks. Therefore, the pattern of mixing costs observed in our previous experiment was not due to the interaction of list switching and task switching.

Our findings also speak to the issue of why the number of lists affected list-switch costs (and task-switch costs) in Experiment 1. In Experiment 2, again list-switch costs tended to be higher when switches between three lists were required than when switching was between two lists. However, this effect vanished after a long CSI. This is precisely what we should expect if removal of no-longer-relevant lists from the DA region takes one to two seconds: With short CSIs, remnants of previously relevant lists will remain in the DA region after a switch, thus contributing to the load on the limited capacity of declarative WM. The increased switch cost with three, as opposed to two, candidate lists could reflect a slow-down of processes under higher load, or additional time that people take to more completely remove the currently irrelevant list(s) from the DA region. With a longer CSI, removal of currently irrelevant lists can be completed during the CSI, so that hardly any trace of them is left in the DA region when the task cue appears.

GENERAL DISCUSSION

We designed a procedure by which the selection of declarative and procedural representations can be investigated under comparable conditions. First, we kept the memory sets and the task sets constant across all trials such that the bindings between retrieval cues and their targets and between the stimuli and responses were fixed for both declarative and procedural representations, respectively. Second, switching between lists and between tasks occurred randomly, independently of each other, and with equal probabilities. Third, we manipulated the number of lists and the number of tasks to be switched among orthogonally. These characteristics enabled us to create a list-switch analogue to the well-established task-switch paradigm.

Our main goal was to test the assumption that declarative WM and procedural WM follow comparable operating principles. We predicted that analogous processing demands would produce analogous empirical effects across these subsystems. In the present study, we focused on the operation of

the central components of declarative and procedural WM: the DA region and the bridge, respectively. As indicated in Table 1, we found evidence for three analogous empirical signatures of the interplay of these components with LTM: list-switch and task-switch costs, preparation effects and residual switch costs, and mixing costs. Additionally, we observed one violation of the analogy between the two subsystems: Whereas it seems that in declarative WM two or more lists could be maintained in the DA region at the same time, procedural WM was highly constrained to a single task set. In what follows, we discuss the evidence for and against analogous operating principles in detail.

Switch costs

In Experiment 1, we observed list-switch costs and task-switch costs of about the same size. In our framework, task-switch costs reflect the time for removing the current task set from the bridge and retrieving the new task set from activated LTM into the bridge. Analogously, list-switch costs reflect the time needed to at least partially remove the current list from the DA region and to retrieve the new list from activated LTM into the DA region. The roughly 200-ms additional time taken for list-switch trials compared to list-repetition trials will not suffice for complete removal of the old list from the DA region, but at least the new relevant list needs to be established with more strength in the DA region than the currently not-relevant list or lists. Achieving a sufficient difference in strength between the new relevant list and remnants of other lists might take longer when traces of two other lists are still lingering in the DA region than when there is only one other list; this could also explain why list-switch costs were larger in the condition with three lists than in the condition with two lists.

Mixing costs

We observed the same pattern of mixing costs in the declarative and procedural subsystems: Requiring the selection of more than one list or task in a

block produced costs, but these costs did not increase in blocks with three lists (or tasks) compared to blocks with only two lists (or tasks). Therefore, one can argue that the mixing costs are not directly related to a higher cognitive load in the mixed blocks than in the single blocks—a load effect would produce continuously increasing mixing costs with increasing number of tasks or lists. In contrast, these findings are more in line with the proposition that mixing costs reflect the need to resolve the ambiguity as to which list, or which task, is relevant on the next trial (Koch, Prinz, & Allport, 2005; Philipp, Kalinich, Koch, & Schubotz, 2008; Poljac, Koch, & Bekkering, 2009; Rubin & Meiran, 2005). Since in mixed blocks the context is ambiguous with respect to which list or task is relevant, the system has to decide in each trial (i.e., both in repetition and in switch trials) which representation to select. In repetition trials, the actual selection of a new representation is not necessary, and therefore the decision time is the same regardless of whether there are two, three, or more lists to select from, and whether there are two or more tasks to select from. In accordance with the ambiguity-resolution account, it has been demonstrated that mixing costs are found only, or are found to be stronger, when task ambiguity is high—for instance, when the task sequence is unpredictable (Poljac et al., 2009)—or when stimuli are bivalent—that is, they can be processed according to both tasks (Koch et al., 2005; Philipp et al., 2008). We venture that mixing costs in declarative WM arise for the same reason: the need to resolve ambiguity as to which list is relevant on the upcoming trial. Thus, we predict that manipulations of the degree of ambiguity, analogous to those carried out in the task-switching literature, also affect list-mixing costs.

Number of sets

We observed at least one violation of the assumed analogy between declarative and procedural WM: Switching between three lists produced higher switch costs than switching between two lists; switching between two versus three task sets, though, had no impact on the task-switch costs.

The effect of number of lists on list-switch costs was found in both Experiments 1 and 2, thus showing that this effect is reliable and not influenced by interactions with the requirement to switch between tasks. As discussed in Experiment 1, these results indicate that when people switch from one list to another, two or more lists (the old and the new candidates for selection) impose a load on the DA region simultaneously for a while (for about 1 to 2 seconds). In contrast, when people switch from one task to another, the old task is removed fairly rapidly and completely, and then the new task is loaded, such that two tasks are not maintained in the bridge simultaneously. Therefore, increasing the number of relevant lists increases the load on the DA region at least temporarily, but no such increase of load occurs in the bridge because it always only holds one task set (see Mayr & Kliegl, 2000).

We ventured that the possible differences between declarative and procedural WM might result from differences in the degree of interference between declarative representations on the one side and the degree of interference between procedural representations on the other side. This possibility deserves further investigation.

Preparation and residual switch costs

In Experiment 2, we manipulated the opportunity to prepare for a list switch by setting up a short or long CSI between the presentation of a list cue and the cue to retrieve and classify the digit. The long CSI reduced list-switch costs to about 85 ms, thus indicating that preparation reduces, but does not eliminate, these costs. Task preparation has also been found to improve performance in task-switch paradigms, but not to completely eliminate the costs associated with switching between tasks (see Kiesel et al., 2010; Monsell, 2003, for reviews). To account for the task preparation effects, some authors proposed two-stage models of task reconfiguration: One stage is related to task configuration and can be initiated upon cue presentation; the other stage refers to response selection processes that have to wait until the target stimulus is presented (e.g., Mayr & Kliegl, 2000; Meiran, 2000;

Rogers & Monsell, 1995). Other authors propose that switch costs reflect interference from the other activated tasks, and residual switch costs reflect sub-optimal preparation for a task switch in some trials (e.g., Altmann & Gray, 2008). All in all, the list preparation and residual list-switch costs obtained in the present study suggest that preparation might exert the same function in list-switch as in task-switch conditions, and that future accounts should consider a broader model that could explain preparatory effects across both domains.

Selection of lists and tasks in WM

We discussed three possible ways in which memory sets and task sets could be selected in WM: (a) serial selection with only one selection step to be carried out at a time; (b) parallel selection in which the memory sets and task sets would be selected at the same time; and (c) composite selection, according to which declarative and procedural representations are unified into an event file that is selected in a single step. The interaction of task switching and list switching enabled a distinction between these three hypotheses. The underadditive interaction was most compatible with the parallel selection hypothesis. Therefore, this result is in good agreement with the assumption of declarative and procedural WM as separate subsystems that can operate in parallel without competing for shared capacity limitations.

CONCLUSION

The observation of similar effects in the selection of declarative and procedural representations supports the assumption of analogous mechanisms for declarative and procedural WM. Nevertheless, the limits on the number of separate sets that can be accommodated in the central component of each these subsystems might differ: More than one memory set can at least temporarily be held in the region of direct access, whereas the bridge seems to be more strictly limited to a single task set. In summary, the assumption of a procedural WM next to a declarative WM seems to be a useful

framework with respect to the investigation of how representations are selected to guide goal-directed reasoning and action.

Original manuscript received 4 March 2011

Accepted revision received 29 July 2011

First published online 15 February 2012

REFERENCES

- Altmann, E. M. (2004). Advance preparation in task switching. *Psychological Science*, *15*, 616–622, doi:10.1111/j.0956-7976.2004.00729.x
- Altmann, E. M., & Gray, W. D. (2008). An integrated model of cognitive control in task switching. *Psychological Review*, *115*, 602–639. doi:10.1037/0033-295X.115.3.602
- Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Mahwah, NJ: Lawrence Erlbaum Associates.
- Baddeley, A. D. (1986). *Working memory*. New York, NY: Oxford University Press.
- Baddeley, A., Chincotta, D., & Adlam, A. (2001). Working memory and the control of action: Evidence from task switching. *Journal of Experimental Psychology: General*, *130*, 641–657. doi:10.1037/0096-3445.130.4.641
- Barrouillet, P., Portrat, S., & Camos, V. (2011). On the law relating processing to storage in working memory. *Psychological Review*, *118*, 175–192.
- Bertelson, P. (1965). Serial choice reaction-time as a function of response versus signal-and-response repetition. *Nature*, *206*, 217–218. doi:10.1038/206217a0
- Brainard, D. H. (1997). The Psychophysics Toolbox. *Spatial Vision*, *10*, 433–436.
- Cowan, N. (1999). An embedded-process model of working memory. In A. Miyake & P. Shah (Eds.), *Models of working memory. Mechanisms of active maintenance and executive control* (pp. 62–101). Cambridge, UK: Cambridge University Press.
- Daneman, M., & Carpenter, P. A. (1980). Individual differences in working memory and reading. *Journal of Verbal Learning and Verbal Behavior*, *19*, 450–466. doi:10.1016/S0022-5371(80)90312-6
- Druey, M. D., & Hübner, R. (2008). Response inhibition under task switching: Its strength depends on the amount of task-irrelevant response activation. *Psychological Research*, *72*, 415–424.

- Garavan, H. (1998). Serial attention within working memory. *Memory & Cognition*, *26*, 263–276.
- Hommel, B., Müsseler, J., Aschersleben, G., & Prinz, W. (2001). The theory of event coding (TEC): A framework for perception and action planning. *Behavioral and Brain Sciences*, *24*, 849–878. doi:10.1017/S0140525X01000103
- Hommel, B., Proctor, R. W., & Vu, K. L. (2004). A feature-integration account of sequential effects in the Simon task. *Psychological Research*, *68*, 1–17. doi:10.1007/s00426-003-0132-y
- Hübner, R., Futterer, T., & Steinhauser, M. (2001). On attentional control as a source of residual shift costs: Evidence from two-component task shifts. *Journal of Experimental Psychology: Learning, Memory & Cognition*, *27*, 640–653. doi:10.1037//0278-7393.27.3.640
- Kessler, Y., & Meiran, N. (2006). All updateable objects in working memory are updated whenever any of them are modified: Evidence from the memory updating paradigm. *Journal of Experimental Psychology: Learning, Memory & Cognition*, *32*, 570–585. doi:10.1037/0278-7393.32.3.570
- Kiesel, A., Steinhauser, M., Wendt, M., Falkenstein, M., Jost, K., Philipp, A. M., et al. (2010). Control and interference in task switching—A review. *Psychological Bulletin*, *136*, 849–874. doi:10.1037/a0019842
- Kiesel, A., Wendt, M., & Peters, A. (2007). Task switching: On the origin of response congruency effects. *Psychological Research*, *71*, 117–125. doi:10.1007/s00426-005-0004-8
- Kleinsorge, T. (1999). Response repetition benefits and costs. *Acta Psychologica*, *103*, 295–310. doi:10.1016/S0001-6918(99)00047-5
- Koch, I., Gade, M., Schuch, S., & Philipp, A. M. (2010). The role of inhibition in task switching: A review. *Psychonomic Bulletin & Review*, *17*, 1–14. doi:10.3758/PBR.17.1.1
- Koch, I., Prinz, W., & Allport, A. (2005). Involuntary retrieval in alphabet-arithmetic tasks: Task-mixing and task-switching costs. *Psychological Research*, *69*, 252–261. doi:10.1007/s00426-004-0180-y
- Kray, J., Li, K. Z. H., & Lindenberger, U. (2002). Age-related changes in task-switching components: The role of task uncertainty. *Brain and Cognition*, *49*, 363–381. doi:10.1006/brcg.2001.1505
- Kübler, A., Murphy, K., Kaufman, J., Stein, E., & Garavan, H. (2003). Co-ordination within and between verbal and visuospatial working memory: Network modulation and anterior frontal recruitment. *NeuroImage*, *20*, 1298–1308. doi:10.1016/S1053-8119(03)00400-2
- Liefvooghe, B., Barrouillet, P., Vandierendonck, A., & Camos, V. (2008). Working memory costs of task switching. *Journal of Experimental Psychology: Learning, Memory & Cognition*, *34*, 478–494. doi:10.1037/0278-7393.34.3.478
- Mayr, U., & Keele, S. W. (2000). Changing internal constraints on action: The role of backward inhibition. *Journal of Experimental Psychology: General*, *129*, 4–26. doi:10.1037//0096-3445.129.1.4
- Mayr, U., & Kliegl, R. (2000). Task-set switching and long-term memory retrieval. *Journal of Experimental Psychology: Learning, Memory & Cognition*, *26*, 1124–1140. doi:10.1037/0278-7393.26.5.1124
- Meiran, N. (1996). Reconfiguration of processing mode prior to task performance. *Journal of Experimental Psychology: Learning, Memory & Cognition*, *22*, 1423–1442. doi:10.1037/0278-7393.22.6.1423
- Meiran, N. (2000). Modeling cognitive control in task-switching. *Psychological Research*, *63*, 234.
- Meiran, N., & Kessler, Y. (2008). The task rule congruency effect in task switching reflects activated long-term memory. *Journal of Experimental Psychology: Human Perception and Performance*, *34*, 137–157. doi:10.1037/0096-1523.34.1.137
- Miyake, A., Emerson, M. J., Padilla, F., & Ahn, J. (2004). Inner speech as a retrieval aid for task goals: The effects of cue type and articulatory suppression in the random task cuing paradigm. *Acta Psychologica*, *115*, 123–142.
- Miyake, A., & Shah, P. (1999). *Models of working memory: Mechanisms of active maintenance and executive control*. New York, NY: Cambridge University Press.
- Monsell, S. (2003). Task switching. *Trends in Cognitive Sciences*, *7*, 134–140. doi:10.1016/S1364-6613(03)00028-7
- Monsell, S., Sumner, P., & Waters, H. (2003). Task-set reconfiguration with predictable and unpredictable task switches. *Memory and Cognition*, *31*, 327–342.
- Oberauer, K. (2001). Removing irrelevant information from working memory: A cognitive aging study with the modified Sternberg task. *Journal of Experimental Psychology: Learning, Memory & Cognition*, *27*, 948–957. doi:10.1037/0278-7393.27.4.948
- Oberauer, K. (2002). Access to information in working memory: Exploring the focus of attention. *Journal of Experimental Psychology: Learning, Memory & Cognition*, *28*, 411–421. doi:10.1037//0278-7393.28.3.411

- Oberauer, K. (2003). Selective attention to elements in working memory. *Experimental Psychology*, *50*, 257–269. doi:10.1027//1618-3169.50.4.257
- Oberauer, K. (2005). Control of the contents of working memory—A comparison of two paradigms and two age groups. *Journal of Experimental Psychology: Learning, Memory & Cognition*, *31*, 714–728. doi:10.1037/0278-7393.31.4.714
- Oberauer, K. (2006). Is the focus of attention in working memory expanded through practice? *Journal of Experimental Psychology: Learning, Memory & Cognition*, *32*, 197–214. doi:10.1037/0278-7393.32.2.197
- Oberauer, K. (2009). Design for a working memory. In B. Ross (Ed.), *The psychology of learning and motivation* (Vol. 51, pp. 45–100). Oxford, UK: Academic Press.
- Oberauer, K., Souza, A. S., Druey, M., & Gade, M. (2011). *Item repetition effects in declarative working memory mirror response repetition effects in procedural working memory*, Unpublished manuscript
- Pashler, H. (1994). Dual-task interference in simple tasks: Data and theory. *Psychological Bulletin*, *116*, 220–244.
- Pelli, D. G. (1997). The VideoToolbox software for visual psychophysics: Transforming numbers into movies. *Spatial Vision*, *10*, 437–442.
- Philipp, A. M., Kalinich, C., Koch, I., & Schubotz, R. I. (2008). Mixing costs and switch costs when switching stimulus dimensions in serial predictions. *Psychological Research*, *72*, 405–414. doi:10.1007/s00426-008-0150-x
- Poljac, E., Koch, I., & Bekkering, H. (2009). Dissociating restart cost and mixing cost in task switching. *Psychological Research*, *73*, 407–416. doi:10.1007/s00426-008-0151-9
- Postle, B. R. (2006). Working memory as an emergent property of the mind and brain. *Neuroscience*, *139*, 23–28.
- Risse, S., & Oberauer, K. (2010). Selection of objects and tasks in working memory. *The Quarterly Journal of Experimental Psychology*, *63*, 784. doi:10.1080/17470210903147486
- Rogers, R. D., & Monsell, S. (1995). Costs of a predictable switch between simple cognitive tasks. *Journal of Experimental Psychology: General*, *124*, 207–231. doi:10.1037/0096-3445.124.2.207
- Rubin, O., & Meiran, N. (2005). On the origins of the task mixing cost in the cuing task-switching paradigm. *Journal of Experimental Psychology: Learning, Memory & Cognition*, *31*, 1477–1491. doi:10.1037/0278-7393.31.6.1477
- Ruchkin, D. S., Grafman, J., Cameron, K., & Berndt, R. S. (2003). Working memory retention systems: A state of activated long-term memory. *Behavioral and Brain Sciences*, *26*, 709–728. doi:10.1017/S0140525X03000165
- Schuch, S., & Koch, I. (2004). The costs of changing the representation of action: Response repetition and response–response compatibility in dual tasks. *Journal of Experimental Psychology: Human Perception and Performance*, *30*, 566–582.
- Steinhauser, M., & Hübner, R. (2005). Mixing costs in task shifting reflect sequential processing stages in a multicomponent task. *Memory & Cognition*, *33*, 1484–1494.